TECHNICAL REPORT RD-GC-88-14

MISSILE ACTUATOR SIMULATION AND AN INVESTIGATION INTO THE ACCURACY OF RUNGE-KUTTA NUMERICAL INTEGRATION

Scott J. Moody
Guidance & Control Directorate
Research, Development, & Engineering Center

JULY 1988

U.S. ARMY MISSILE COMMAND
Redstone Arsenal, Alabama 35898-5000

Approved for public release; distribution is unlimited.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188
Exp. Date: Jun 30, 1986

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) RD-GC-88-14	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Guidance & Control Directorate Res, Dev, & Eng (RD&E) Center 6b. OFFICE SYMBOL (If applicable) AMSMI-RD-GC-N	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Commander U.S. Army Missile Command, ATTN: AMSMI-RD-GC Redstone Arsenal, AL 35898-5254	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
MISSILE ACTUATOR SIMULATION AND AN INVESTIGATION INTO THE ACCURACY OF RUNGE-KUTTA NUMERICAL INTEGRATION

12. PERSONAL AUTHOR(S)
Scott J. Moody

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM ______ TO Apr 88	JULY 1988	51

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report discusses the computer simulation of the ATACMS fin actuators. These subroutines are one of many possible methods of representing actuator behavior in a 6DOF (six degree of freedom) or 5DOF missile simulation. Also included is a section to discuss the accuracy of a version of the Runge-Kutta integration method. This particular simulation was run on a Zenith personal computer with Ryan-McFarland FORTRAN.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Scott J. Moody	22b. TELEPHONE (Include Area Code) (205) 876-1532 22c. OFFICE SYMBOL AMSMI-RD-GC-N

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

EXECUTIVE SUMMARY

A linear second-order system was implemented in the simulation in a preliminary attempt to model the physical actuator's dynamics. The second-order system with non-linearities added was tried next. However, we presently have no data on the actual performance of the actuator and therefore cannot assess the validity of our models. The integration scheme gave very good accuracy and overall performance when a step size of 0.001 seconds was chosen.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

I. INTRODUCTION

Missile simulation is an indispensable tool with which the engineer's computer can emulate the behavior of an actual missile in flight. Because a simulation is usually not an exact reproduction of a physical system, the engineer must make certain assumptions or approximations as to how a physical system should be mathematically represented in a computer program. The results of the simulation should allow the engineer to verify the performance of the system and to validate his/her assumptions.

The most straightforward method of tackling a missile system simulation would be to divide the system into discrete sections and construct subroutines that would accurately model their tangible equivalent. The subroutines that will be addressed in this report are the fin actuators and the numerical integration scheme.

II. FIN ACTUATOR MODELS

Two types of fin actuator models that we will be discussing are a linear second-order model and a non-linear second-order model.

A. Linear Model

A simple model that could describe an actuator's dynamics is a linear second-order system with damping zeta (ζ) and natural frequency omega (ω_n). Such a system exemplifies unity gain out to ω_n, peaking of gain at ω_n inversely proportional to damping, a -20dB per decade slope in gain after ω_n, and a 180° change in phase. Step response characteristics such as rise time and overshoot are a function of bandwidth (ω_n) and damping.

The transfer function of a second-order system is given below, where δ is the output and δ_c is the input. Figure 1 shows one of many possible methods of implementing the transfer function as a block diagram.

$$G(s) = \frac{\delta}{\delta_c} = \frac{\omega_n^2}{S^2 + 2\zeta\omega_n s + \omega_n^2}$$

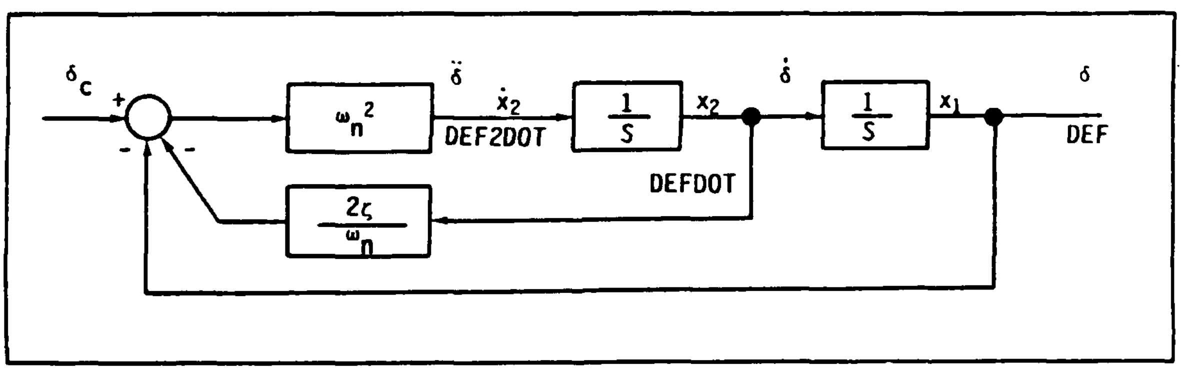


Figure 1. Second-order block diagram.

The differential equation describing the system's dynamics is:

$$\ddot{\delta} = \omega_n^2 \left(\delta_c - \delta - \frac{\dot{\delta} 2\zeta}{\omega_n} \right)$$

The differential equation when encoded in FORTRAN becomes:

```
DEFDD = (OMEGA**2)*(DEFC-DEF-(DEFD*2*ZETA/OMEGA))
```

The system's state equations are as follows:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\zeta\omega_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \omega_n^2 \end{bmatrix} \delta_c$$

$$\delta = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Where δ (x_1) is the fin position, $\dot{\delta}$ (x_2) is the fin velocity, and $\ddot{\delta}$ ($\dot{x}_2$) is the fin acceleration.

1. Analytical Solution

Since there will be differences between the solutions to the differential equations for the actual and simulated systems, we will develop an analytical solution for comparison. The input (δ_c) will be a unit step.

$$\delta_c(t) = 1$$

$$\delta_c(S) = \frac{1}{S}$$

from the transfer function:

$$\delta(S) = G(S) \cdot \delta_c(S) = \frac{\omega_n^2}{S(S^2 + 2\zeta\omega_n S + \omega_n^2)}$$

$$\delta(t) = L^{-1}[\delta(S)] \quad ..$$

From a table of Laplace transform pairs, we find the solution to $\delta(t)$:

$$\delta(t) = \left[\frac{1}{\omega_n^2} - \frac{1}{\omega_n^2 \sqrt{1 - \zeta^2}} e^{-\zeta\omega_n t} \sin(\omega_n \sqrt{1-\zeta^2} + \cos^{-1}\zeta) \right] \omega_n^2$$

$$\delta(t) = 1 - \frac{e^{-\zeta\omega_n t}}{\sqrt{1-\zeta^2}} \sin(\omega_n \sqrt{1 - \zeta^2} + \cos^{-1}\zeta)$$

For a solution to the first integral, $\dot{\delta}(t)$:

$$\dot{\delta}(t) = L^{-1}[S\,\delta(S) - \delta(0+)]$$

assume $\delta(0+) = 0$.

$$S\,\delta(S) = \frac{\omega_n^2}{S^2 + 2\zeta\omega_n S + \omega_n^2}$$

From the Laplace transform pairs table:

$$\dot{\delta}(t) = \omega_n^2 \left[\frac{1}{\omega_n\sqrt{1-\zeta^2}} e^{-\zeta\omega_n t} \sin(\omega_n\sqrt{1-\zeta^2}\,t) \right]$$

$$\dot{\delta}(t) = \frac{\omega_n e^{-\zeta\omega_n t} \sin(\omega_n\sqrt{1-\zeta^2}\,t)}{\sqrt{1-\zeta^2}} \quad .$$

Using an ω_n of 144 rads/sec, and a ζ of 0.6, as an example we get:

$$\delta(t) = 1-1.25\, e^{-86.4t} \sin(115.2t + 0.9273) \text{ rad}$$

$$\dot{\delta}(t) = 180\, e^{-86.4t} \sin(115.2t) \text{ rad/sec} \quad .$$

2. Simulation Results

Output data from the simulation were tabulated and plotted to produce the graphs shown in Figures 2, 3, and 4. Figure 2 shows fin position (angle) versus time. Figure 3 is a plot of fin velocity versus time and Figure 4 shows fin acceleration versus time. These outputs are in response to a step input. Comparison of analytical and simulated solutions are given in Section III. The FORTRAN code for this program and a short discussion are given in the Appendix.

B. Non-Linear Model

A second type of model is one that contains physical limitations, which were added to the linear second-order model to yield a second-order non-linear model.

1. Development

The second-order linear model was modified to include characteristics typical of an actuator motor. This resultant non-linear model more closely emulates the real thing. These characteristics are inherent limitations of the physical system and are non-linear. They include; position limits (fin stops), velocity limits (slew rate limits), acceleration limits (finite torque), a deadband in the rate feedback (hysteresis), and aerodynamic hinge moments. Figure 5 shows a block diagram with the note describing the differential equation.

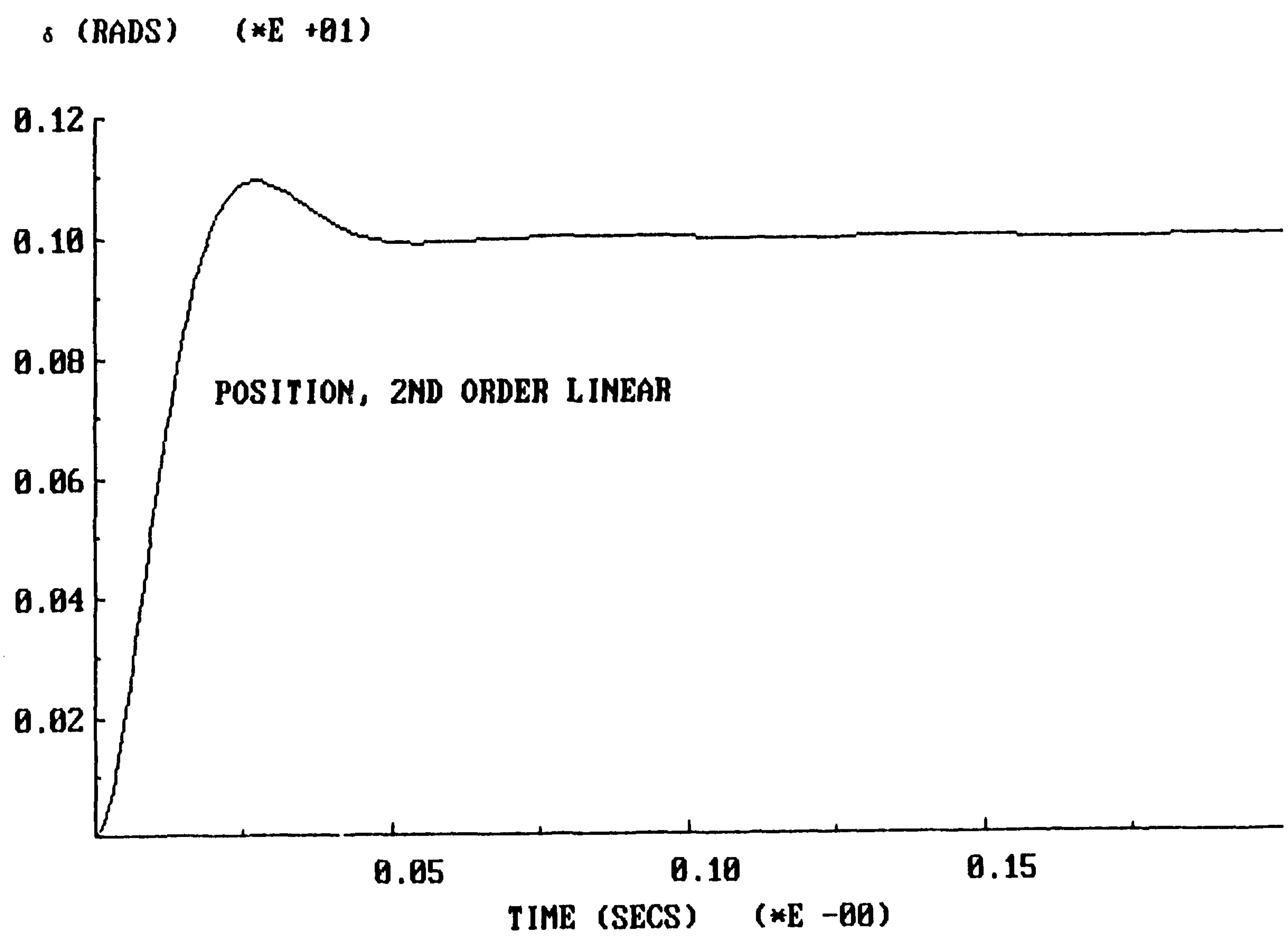


Figure 2. Plot of fin angle versus time for the linear model.

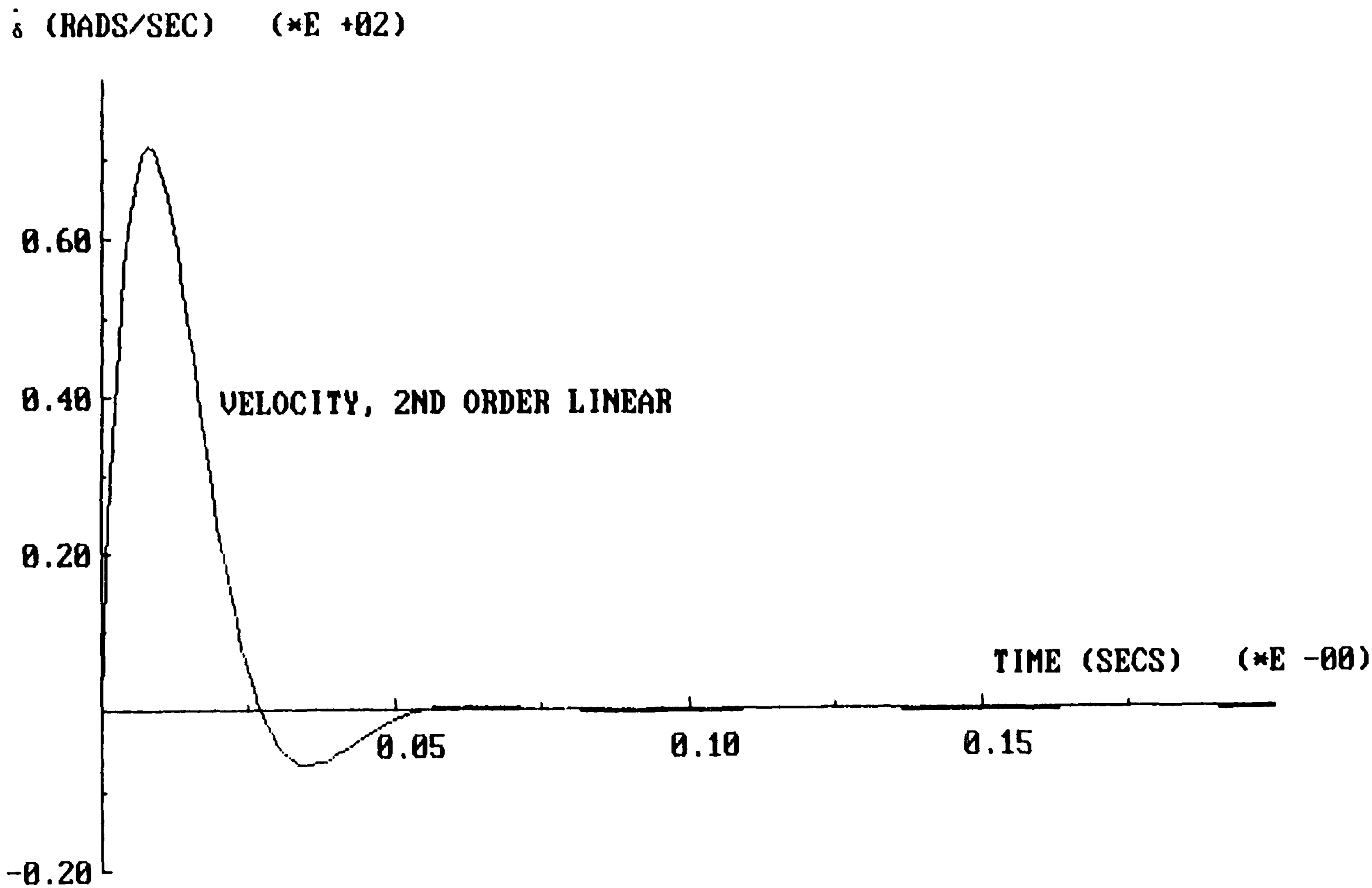


Figure 3. Plot of fin velocity versus time, linear model.

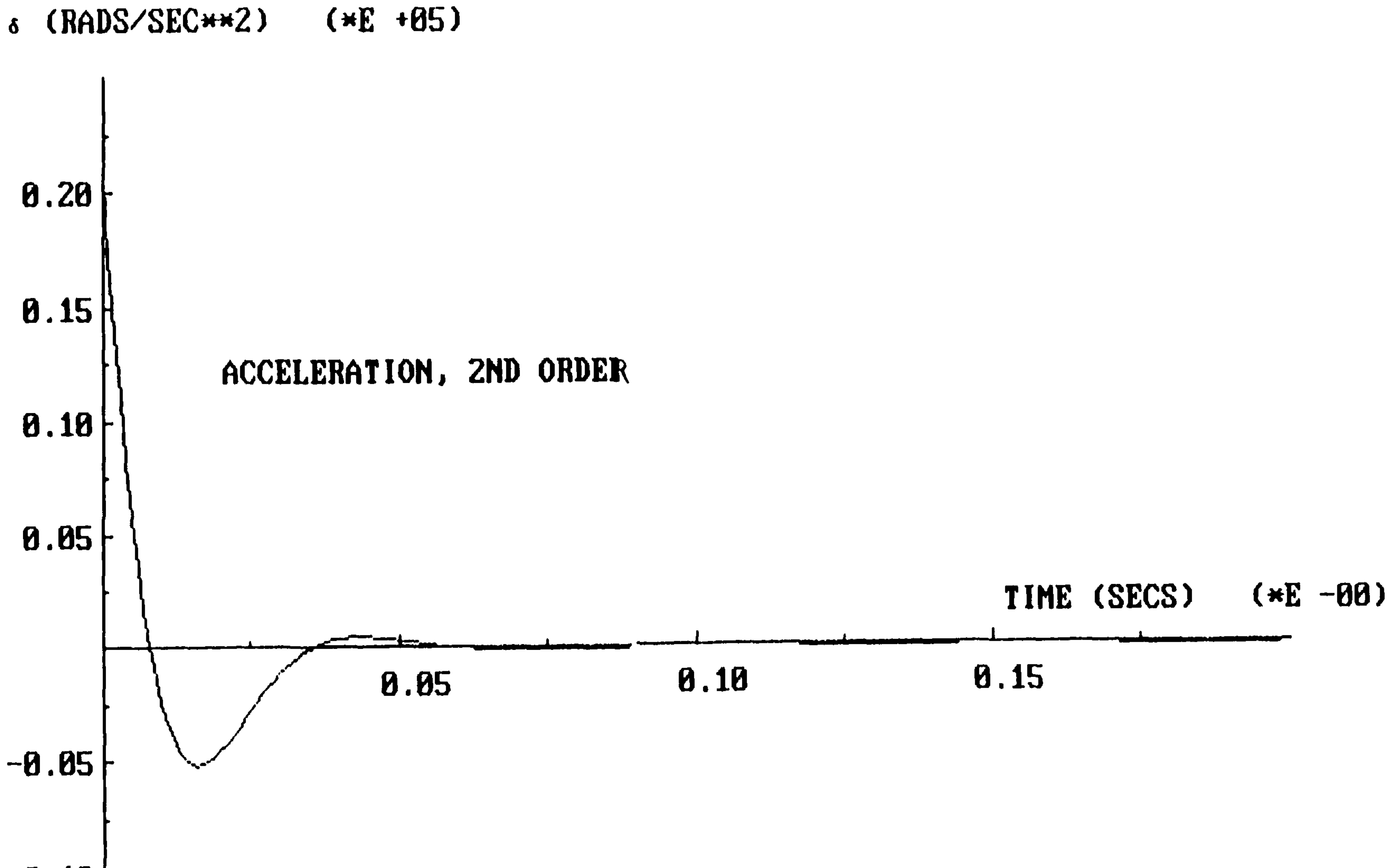


Figure 4. Plot of fin acceleration versus time, linear model.

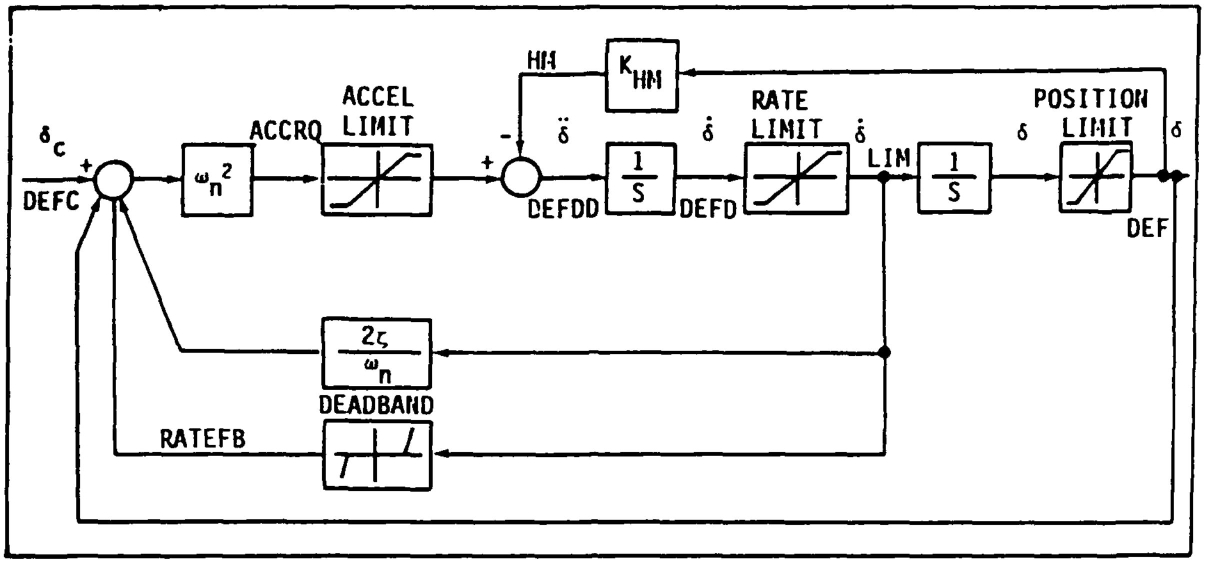


Figure 5. <u>Second order non-linear block diagram</u>.

NOTE: Descriptive differential equation is:

$$\ddot{\delta} = \omega_n^2 \left(\delta_c - \delta_{LIM} - \text{RATEFB} - \dot{\delta} \frac{2\zeta}{\omega_n} \right) - \text{HM}$$

The describing differential equation that appears in the simulation code is:

DEFDD = (OMEGA**2)*(DEFC - DEF - RATEFB - (DEFD*2.*ZETA/OMEGA)) - HM

where: DEF = δ = fin position

DEFD = $\dot{\delta}$ = fin velocity

DEFDD = $\ddot{\delta}$ = fin acceleration

HM = hinge moment

RATEFB = rate feedback

OMEGA = bandwidth (ω_n)

ZETA = damping factor.

As can be seen from the block diagram, no trivial analytical solution for the differential equation exists, so we must rely on a numerically integrated solution.

2. Simulation results

The fin angle, velocity, and acceleration versus time for the non-linear second-order model were plotted and appear in Figures 6, 7, and 8. The hinge moment for these plots was set to zero.

Compared to the linear model, the position rise time is lengthened due to the rate and acceleration limits of 5.25 rads/sec and 300 rads/sec^2, respectively. The overshoot is less due to the position limit of 0.436 rads. Plots for the negative fin deflection were not given since the results were a mirror image of the positive deflection.

Various hinge moment constants were added and the results are shown in Figures 9 through 12. The hinge moment in Figure 9 is an aiding moment: Note the increased overshoot and faster rise time. The hinge moments in Figures 10 through 12 are hindering moments: Note that some of the plots show that the commanded fin deflections were not achieved and in Figure 12 a limit cycle resulted. The complete FORTRAN code for this non-linear model is included in the Appendix.

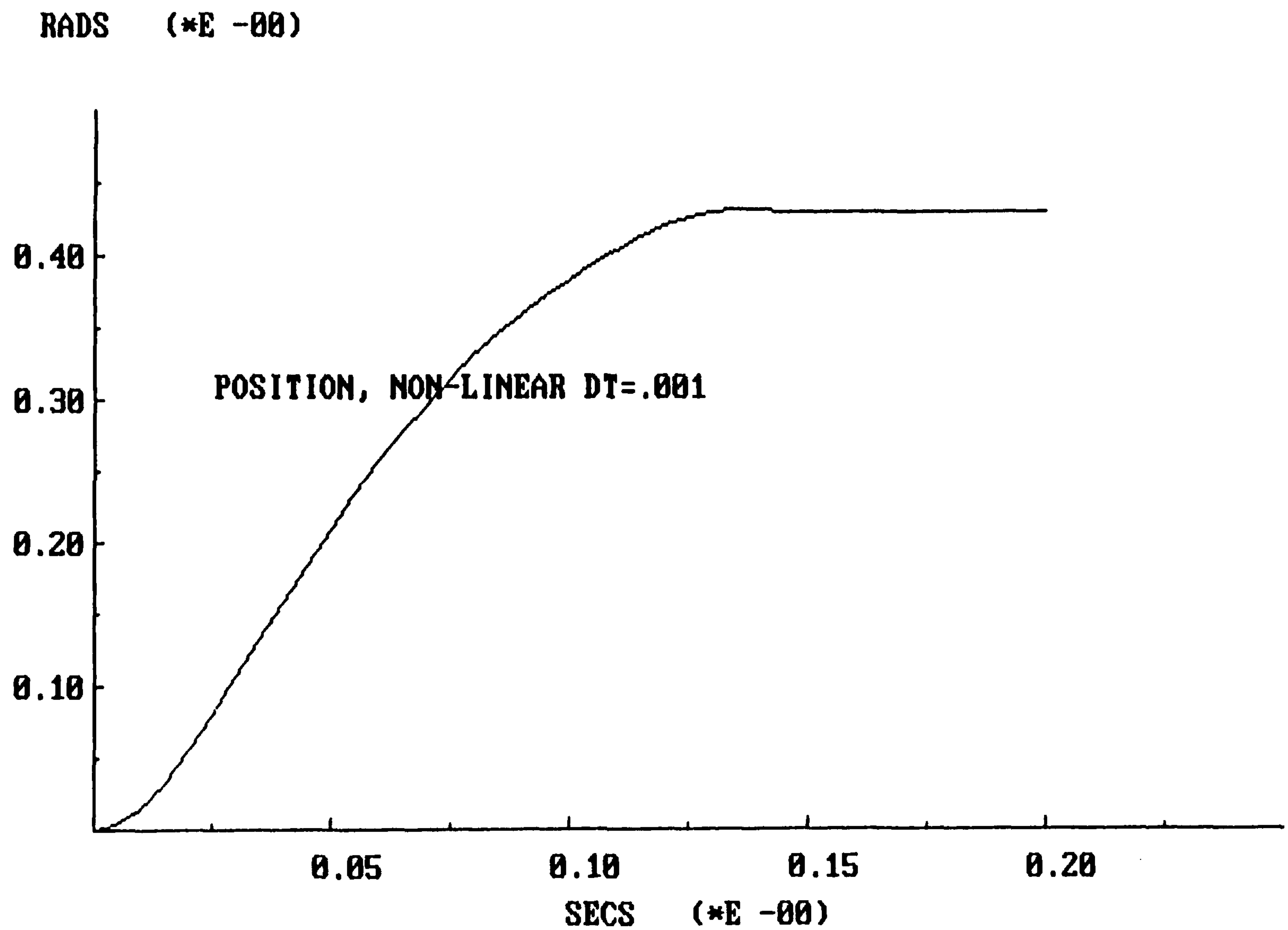


Figure 6. Plot of fin position vs. time, non-linear.

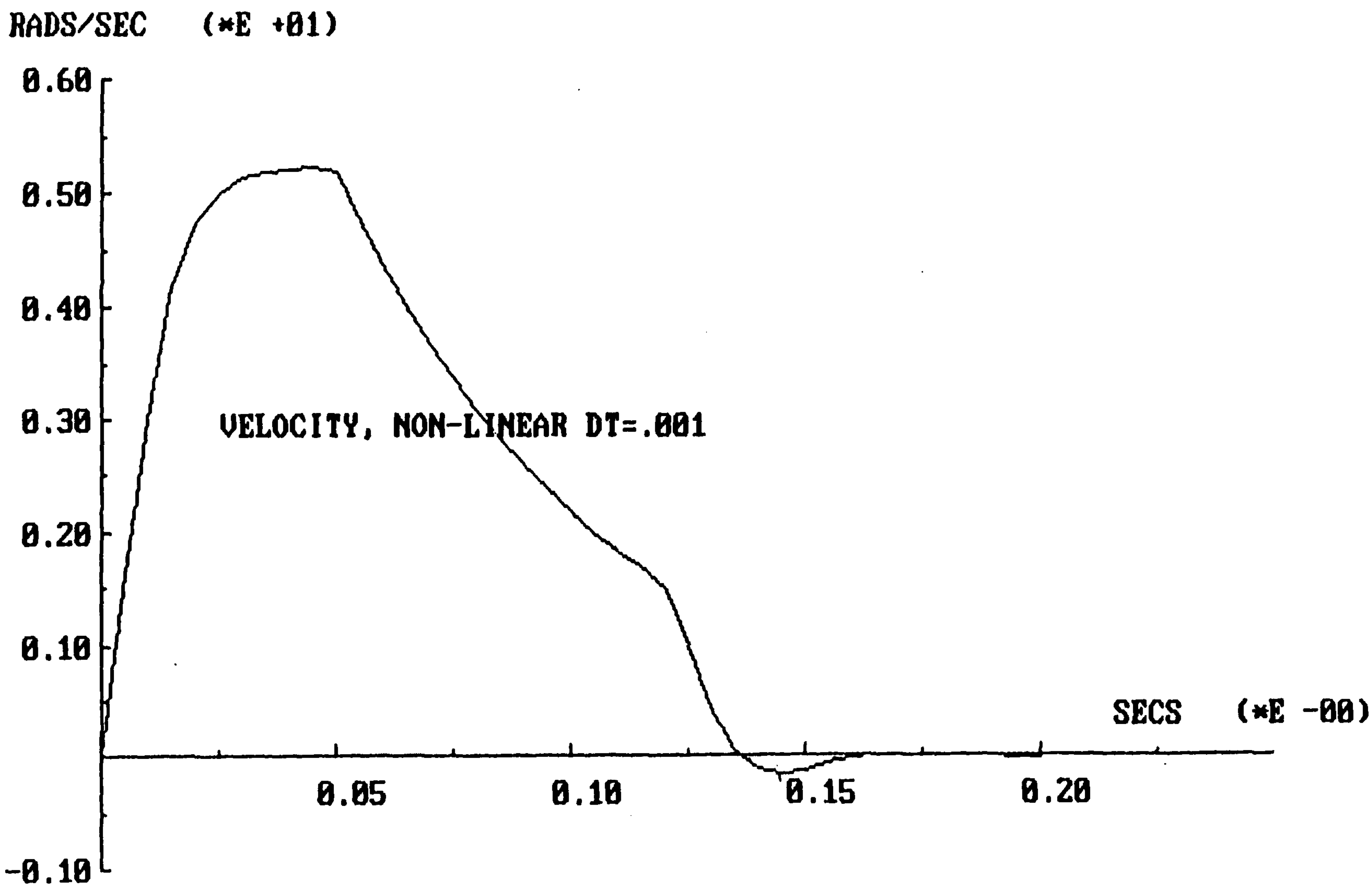


Figure 7. Plot of fin velocity vs. time, non-linear.

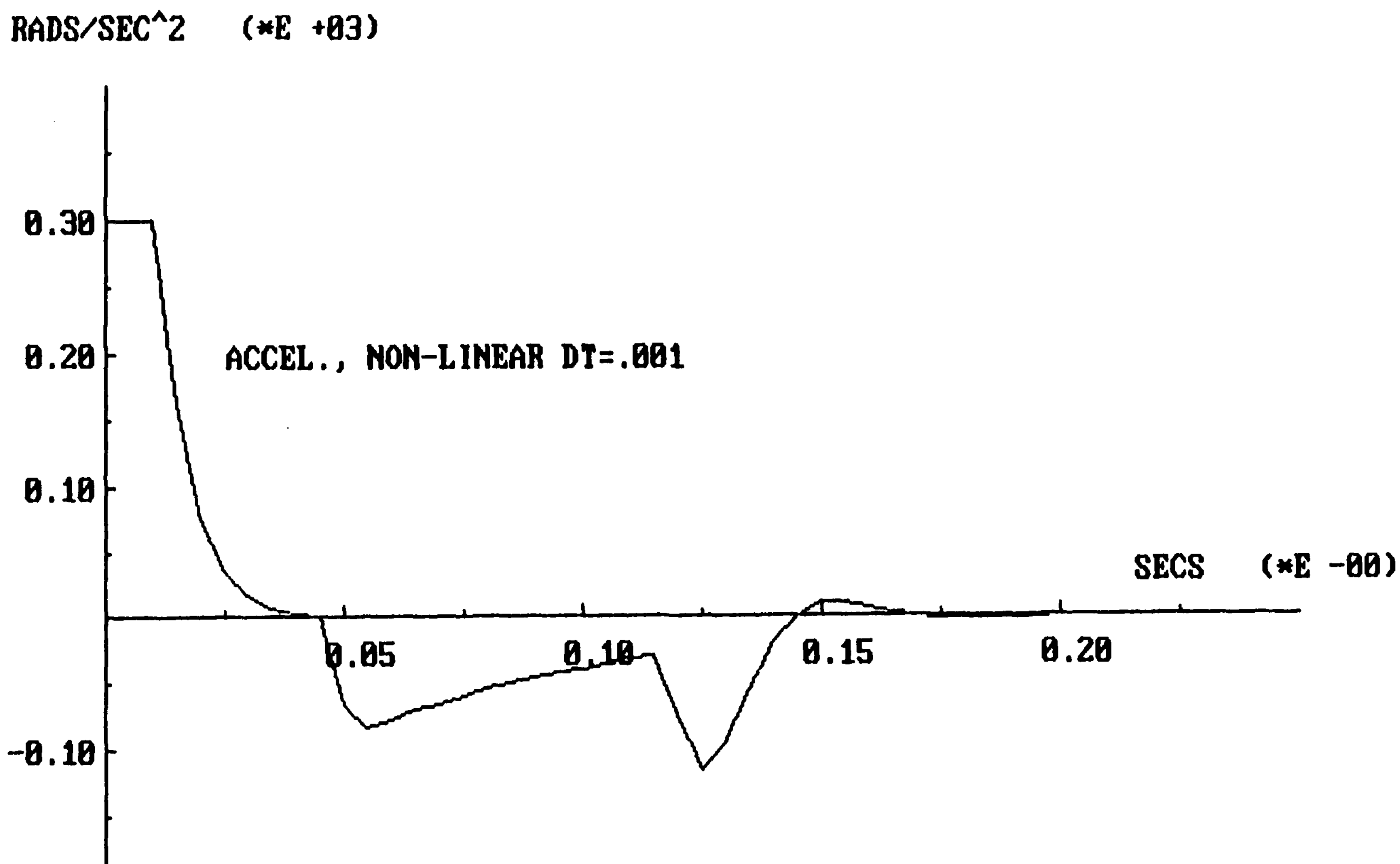


Figure 8. Plot of fin acceleration vs. time, non-linear.

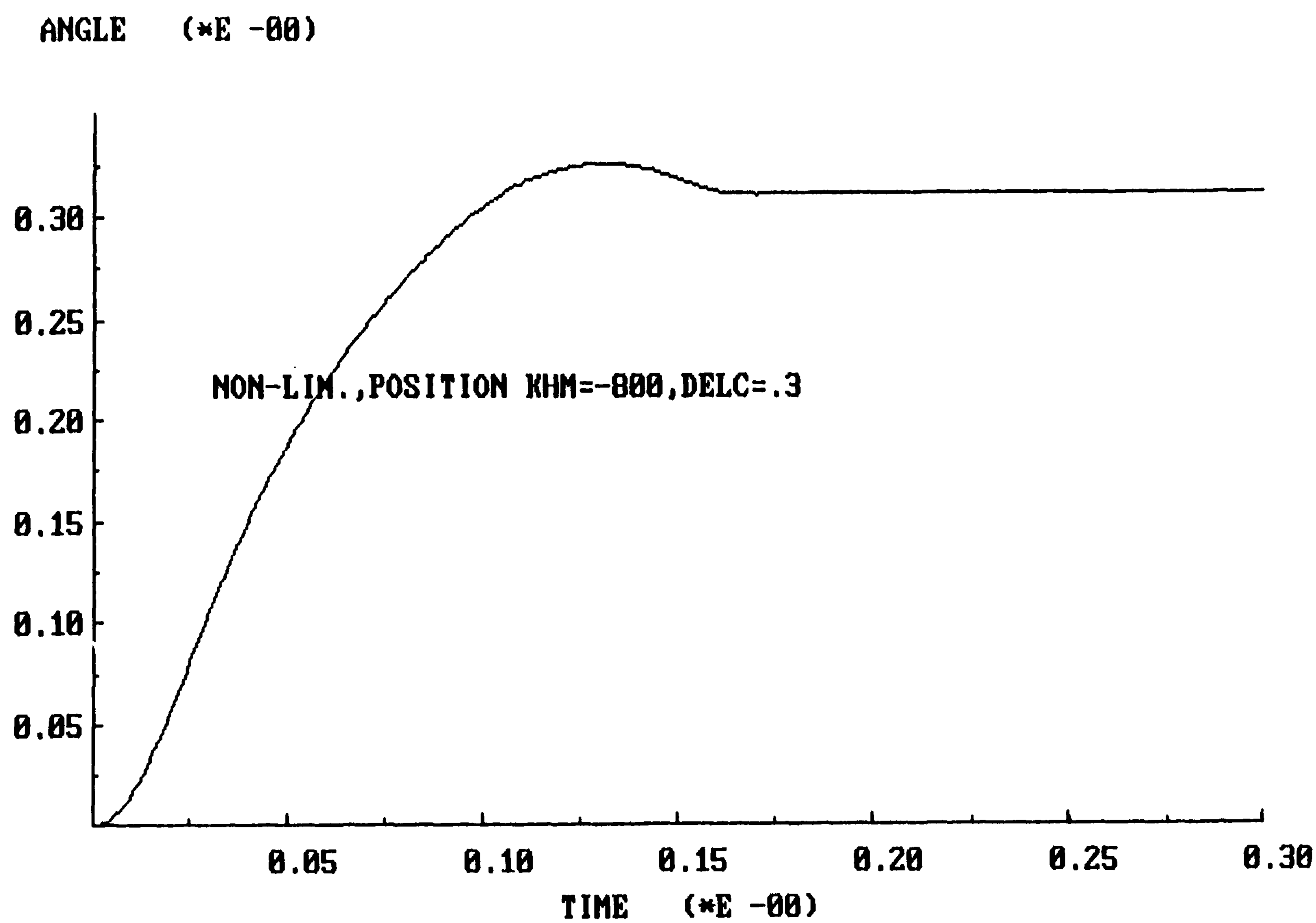


Figure 9. Plot of fin angle vs. time, hinge moment k = -800.

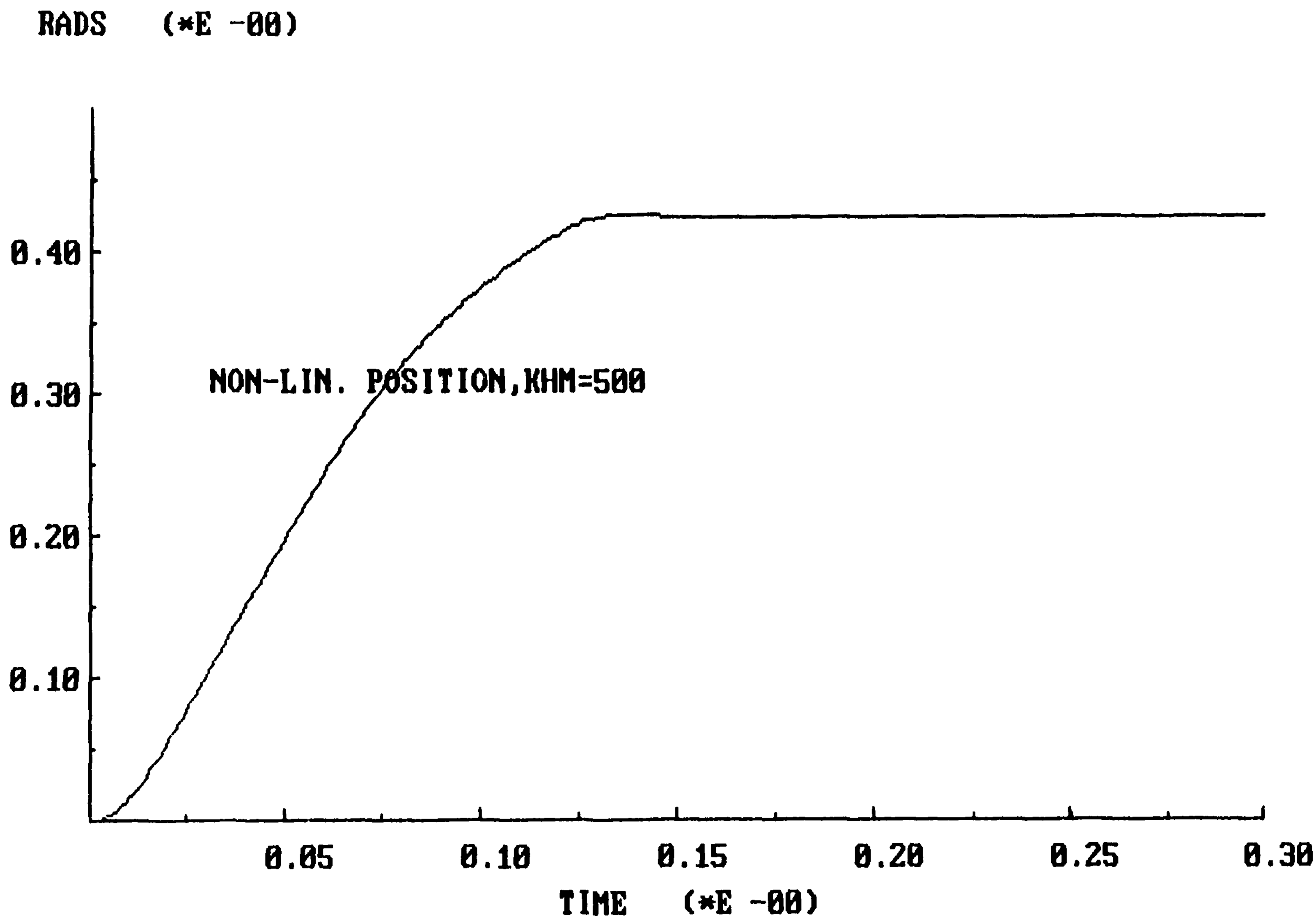


Figure 10. Plot of fin angle vs. time, hinge moment k = 500.

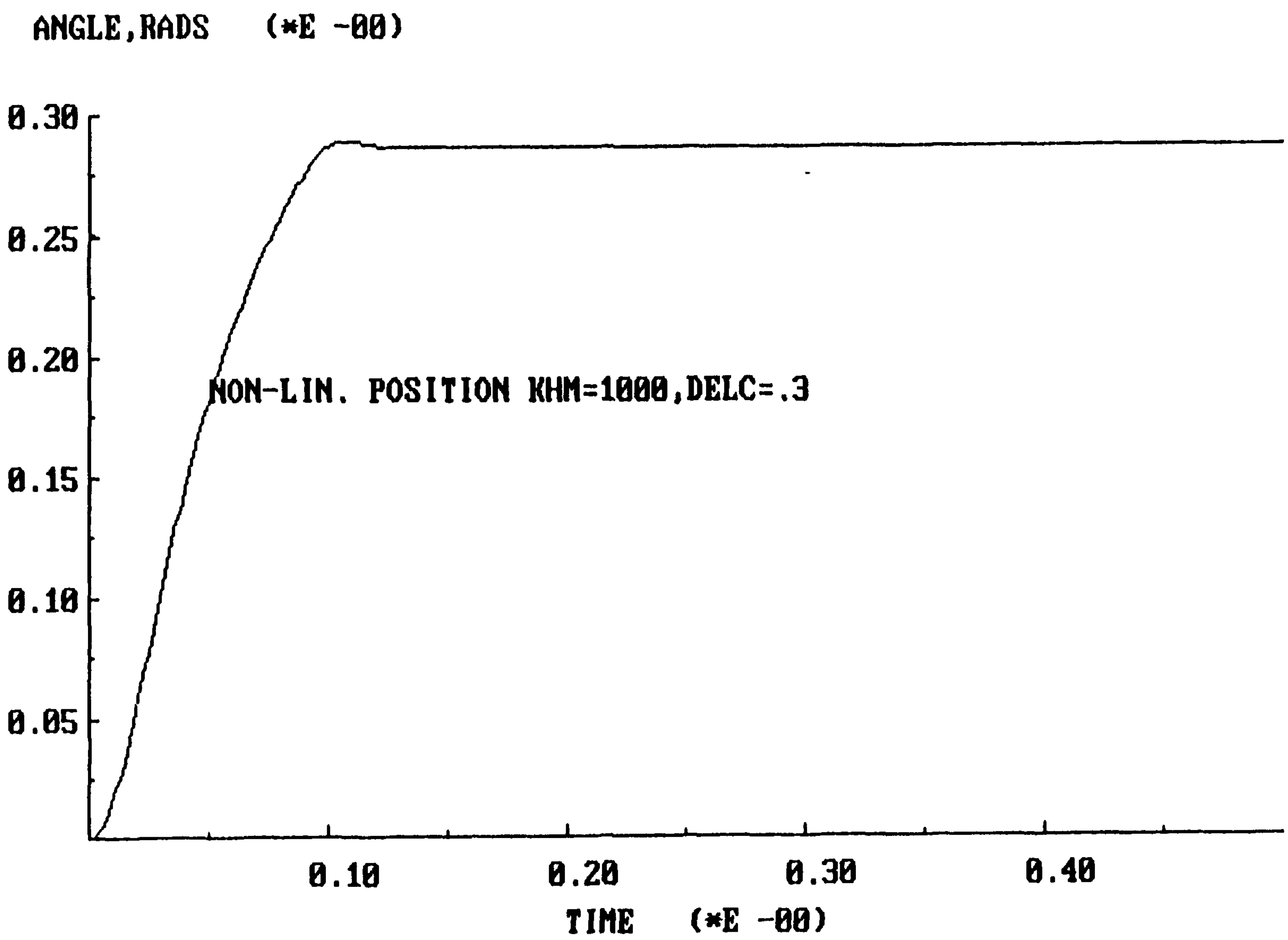


Figure 11. Plot of fin angle vs. time hinge moment k = 1000.

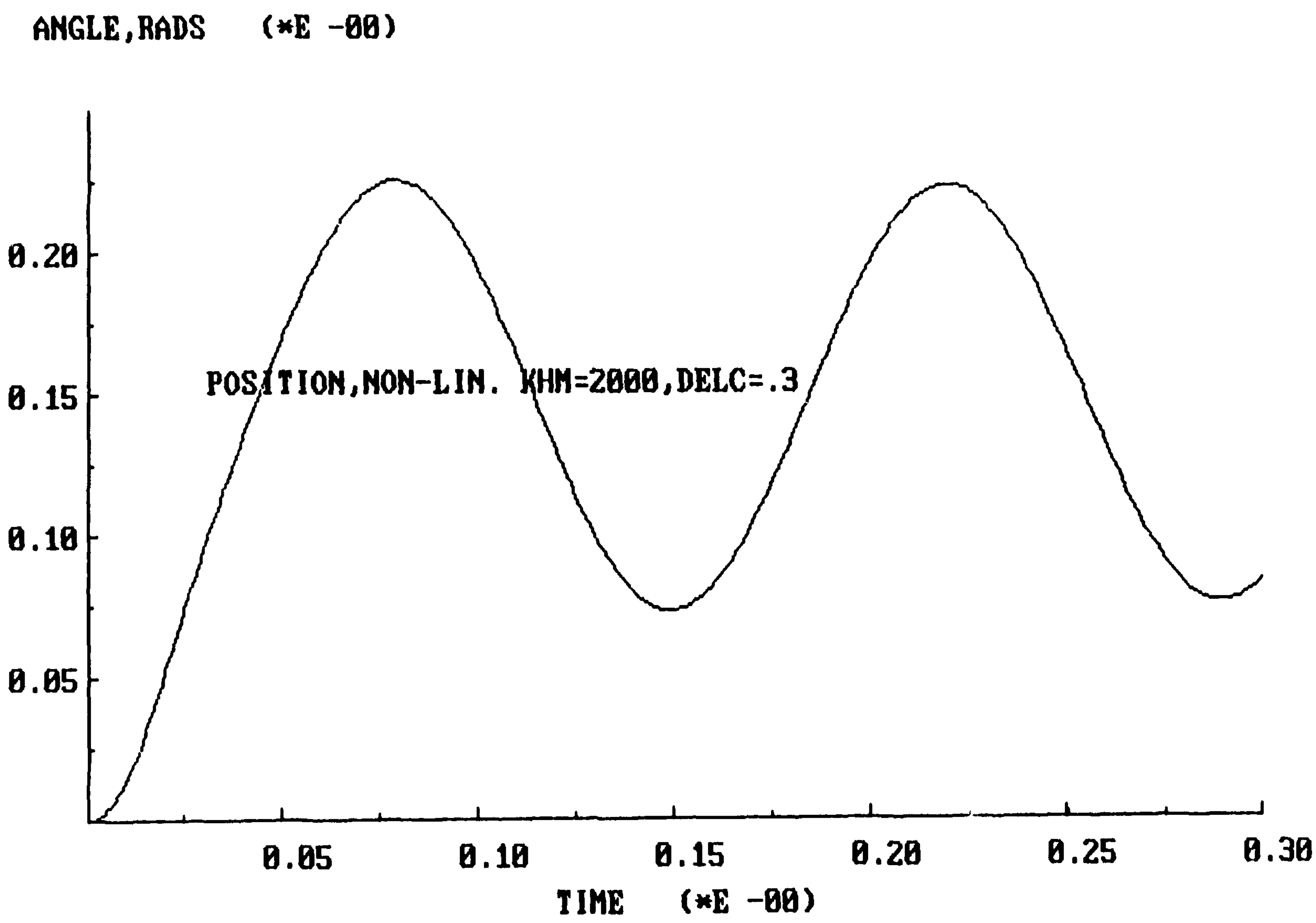


Figure 12. Plot of fin angle vs. time, hinge moment k = 2000.

III. NUMERICAL INTEGRATION

A method of solving differential equations in a computer program is to find the solutions by numerical integration. This process usually involves iterative mathematical calculations.

A. Development

Solutions to mathematically modeled systems with transient or time-dependant processes usually involves solving differential equations. Some differential equations can be solved analytically, however most cannot, particularly if non-linear elements are present. Therefore, in computer programs, numerical integration methods can be used to solve differential equations with acceptable accuracy.

One numerical integration method commonly used is the Runge-Kutta fourth-order method (R-K 4). There are many good texts available that investigate and develop this method in detail, however, only a general overview will be given here.

The R-K 4 scheme uses a weighted average of four estimates to calculate an approximation to the solution. A simple form of a first order differential equation is:

$$\frac{dx}{dt} = f(x,t) \quad ,$$

where $f(x,t)$ is a known function. Substituting values xn and tn into the equation we get the slope of the solution curve at a known starting point, tn.

The R-K 4 approach is to find an approximate slope at a known starting (xn,tn) and use this slope and a small time increment to proceed to the next point. Then assuming that this new point is a known point on the solution line, again find an approximate slope at this new point and proceed to the next point by incrementing the time step. The equations are:

$$k_1 = f(tn,xn)$$

$$k_2 = f(tn + \frac{h}{2}, xn + \frac{k_1 h}{2})$$

$$k_3 = f(tn + \frac{h}{2}, xn + \frac{k_2 h}{2})$$

$$k_4 = f(tn + h, xn + k_3 h)$$

$$xn{+}1 = xn + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

where: h is the time increment
tn is the initial time
xn is the initial solution point.

B. Accuracy

The accuracy of the solution is a function of the time step, h. A smaller time step results in better accuracy but the program run times are increased compared to a larger time step. Too small a time step can cause round-off or truncation errors due to the increased number of calculations. There is an optimum range, for each application, of time step values for the integration scheme.

1. Analytical Solutions

To check the accuracy of the integration scheme as a function of time step, the analytical solution to a second-order linear differential equation was compared to a R-K 4 solution. The linear second-order actuator model developed earlier will be used as an example. Various time steps were used and the resultant average and maximum percent errors were tabulated and plotted. The fin position and velocity (δ and $\dot{\delta}$) were evaluated using the differential equations and analytical solutions presented earlier. The simulation was performed using a Zenith personal computer with an 80286 central processor and an 80287 numeric data processor, with all calculations done in double precision. The describing differential equation given earlier is:

$$\ddot{\delta} = \omega_n{}^2 \delta_c - \delta\omega_n{}^2 - 2\zeta\omega_n\dot{\delta}$$

with a unit step input ($\delta_c(t) = 1$) .

This equation will be presented to the R-K 4 integrator to arrive at a solution of the first integral, velocity $\dot{\delta}$. The analytical solution is:

$$\dot{\delta}(t) = 180e^{-86.4t} \sin(115.2t) \text{ rad/sec} \quad .$$

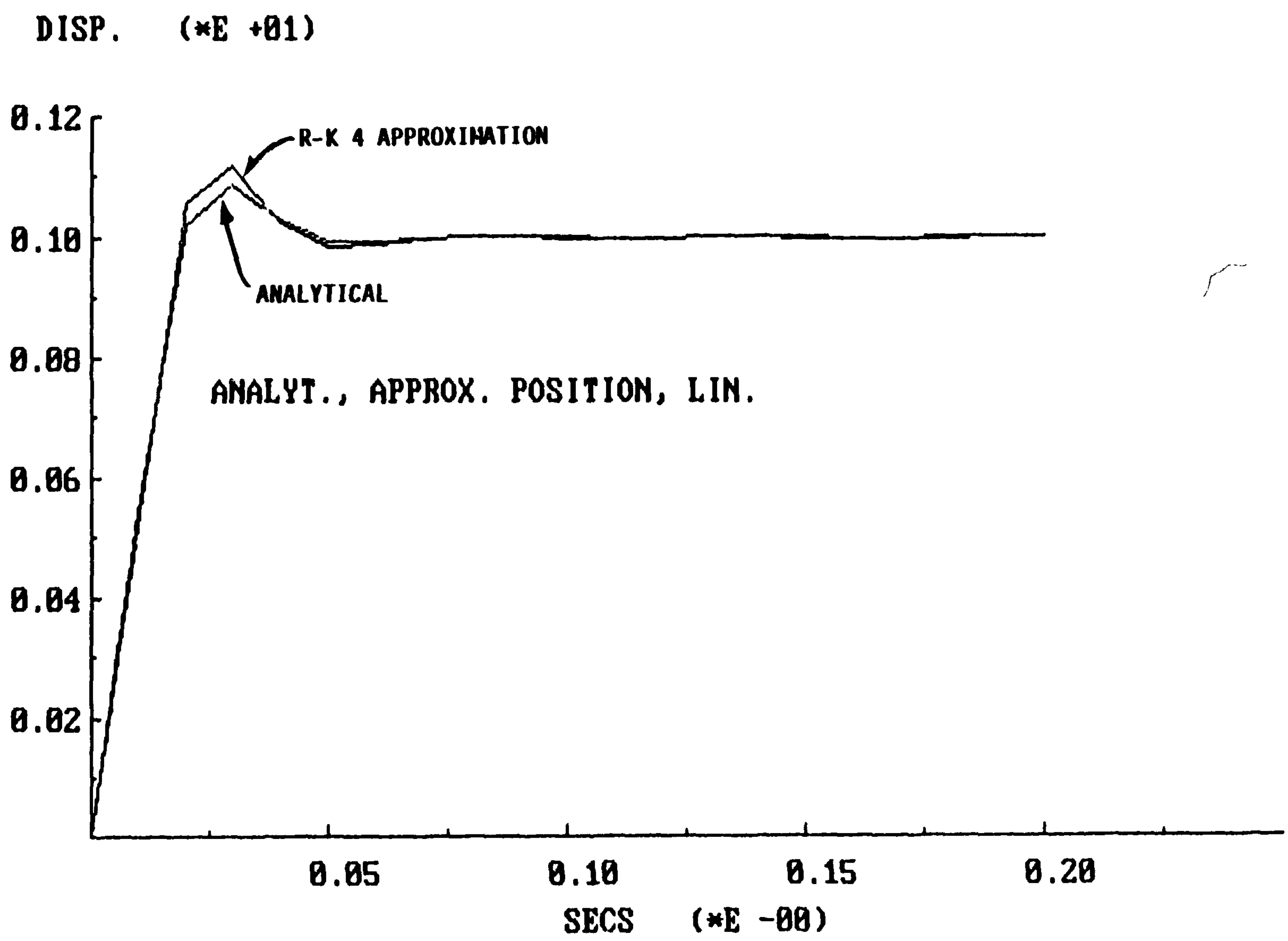
The first integral, $\dot{\delta}$, will again be presented to the R-K 4 integrator to calculate the second integral δ, position (fin angle). The analytical solution is:

$$\delta(t) = 1\text{-}1.25e^{-86.4t} \sin(115.2t + 0.9273) \text{ rad} \quad .$$

2. Simulation Results

The analytical solutions and the R-K 4 approximations for both position and velocity were put on the same plot and are shown in Figures 13 and 14. Note the visible errors in the two plots.

The time step for these plots was coarse, 0.01 secs, and took a few seconds to complete. The percent errors at each time interval were averaged and are given for position and velocity as well as the maximum percent errors (see Table 1).



DT = 0.01 sec

Figure 13. Plot of numerical and analytical solutions to fin angle vs. time, time step = 0.01 seconds.

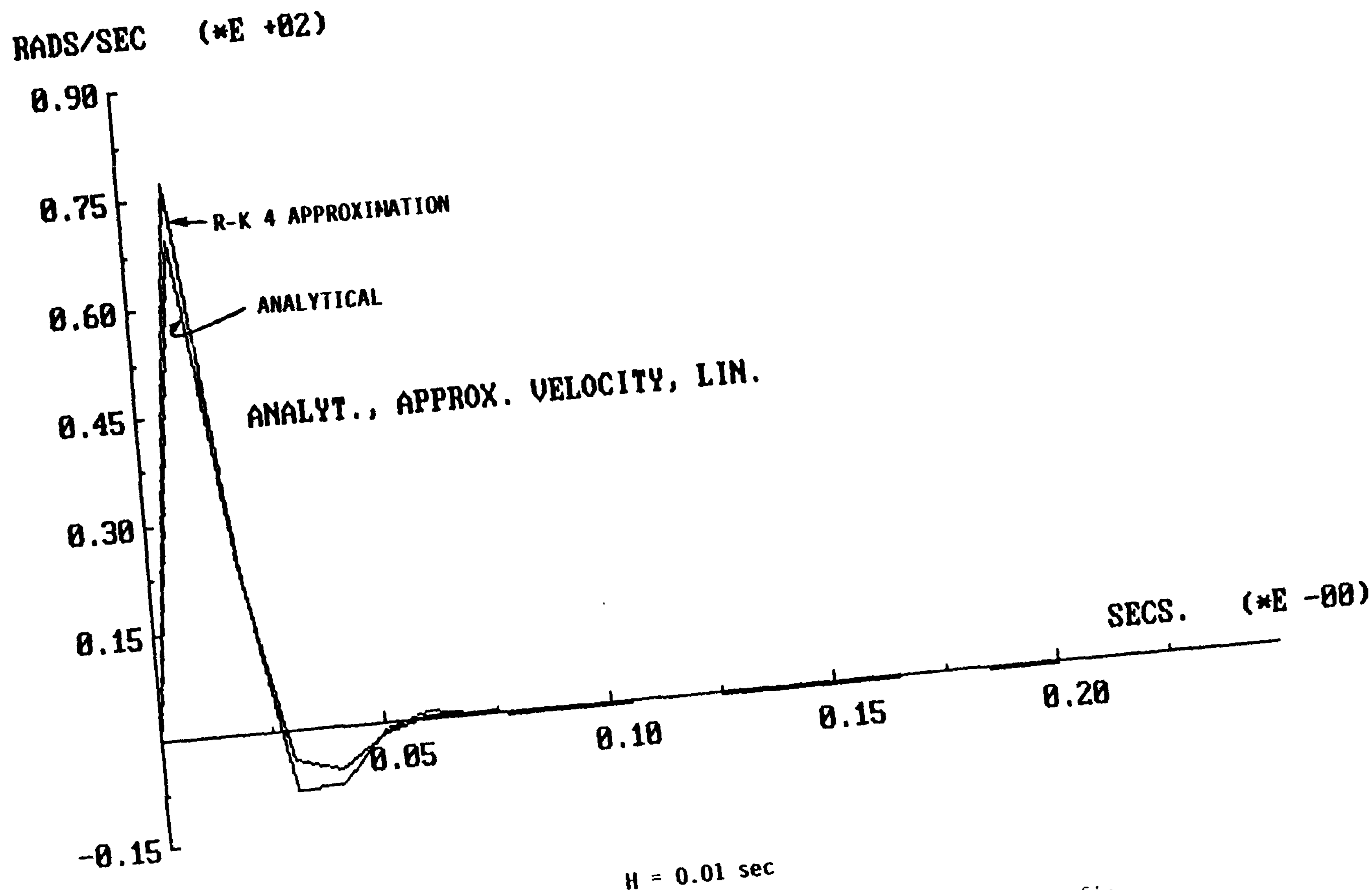


Figure 14. Plot of numerical and analytical solutions to fin velocity vs. time, time step = 0.01 seconds.

TABLE 1. Table of Average and Maximum Percent Errors, $h = 0.01$ secs

	Average Percent	Maximum Percent
Position, δ	0.0511	3.96
Velocity, $\dot{\delta}$	69.6	4000

Next, a smaller time step of 0.001 seconds was used. The errors are given in Table 2 and the results are plotted in Figures 15 and 16. Note that there are no visible differences in the analytical and numerically integrated solutions.

TABLE 2. Table of Average and Maximum Percent Errors, $h = 0.001$ secs

	Average Percent	Maximum Percent
Position, δ	3.2×10^{-5}	4.7×10^{-3}
Velocity, $\dot{\delta}$	1.9×10^{-3}	0.21

A time step of 0.0001 seconds was used and the plots are not shown because the errors were not visible, however the errors are listed in Table 3.

TABLE 3. Table of Average and Maximum Percent Errors, $h = 0.0001$ secs.

	Average Percent	Maximum Percent
Position, δ	2.4×10^{-4}	5.7×10^{-3}
Velocity, $\dot{\delta}$	2.7×10^{-7}	5.9×10^{-5}

Using a stop time of 0.1 seconds and a resolution of 0.0005 seconds, the average percent errors for the first and second integrals were plotted as a function of step size in Figures 17 and 18. The maximum percent errors versus step size were plotted in Figures 19 and 20. Note that initially, a large step size results in poor accuracy. Smaller step sizes give better accuracy, as would be expected, but too small a step size causes the errors to increase. An optimum step size can be chosen from these graphs or from similar graphs from other applications.

The simulation was run on a VAX 11/780 to investigate possible wordlength effects on the results; however, the precision for the two machines are the same and similar results were obtained.

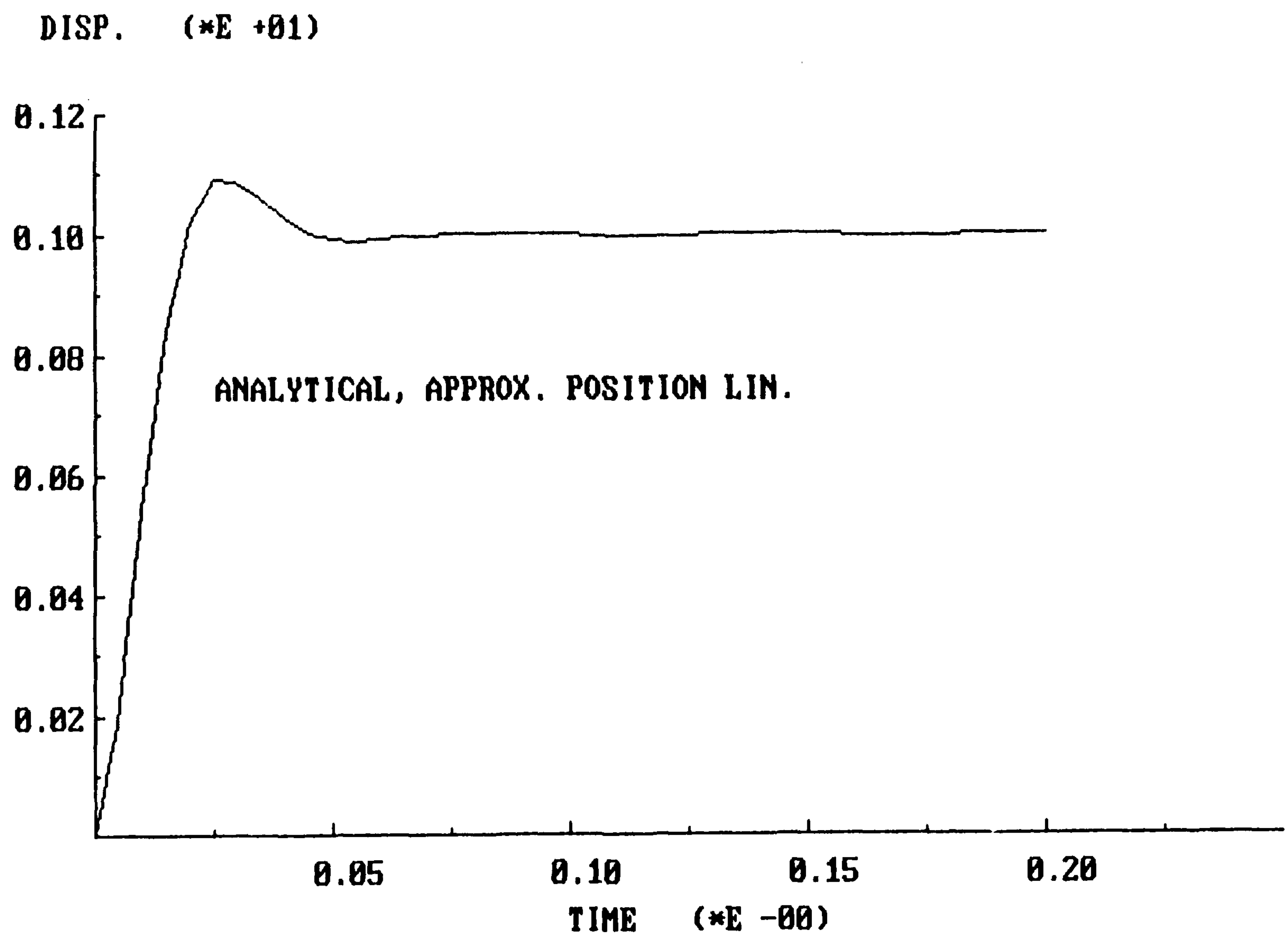


Figure 15. Plot of numerical and analytical solutions to fin angle vs. time, time step =0.001 seconds.

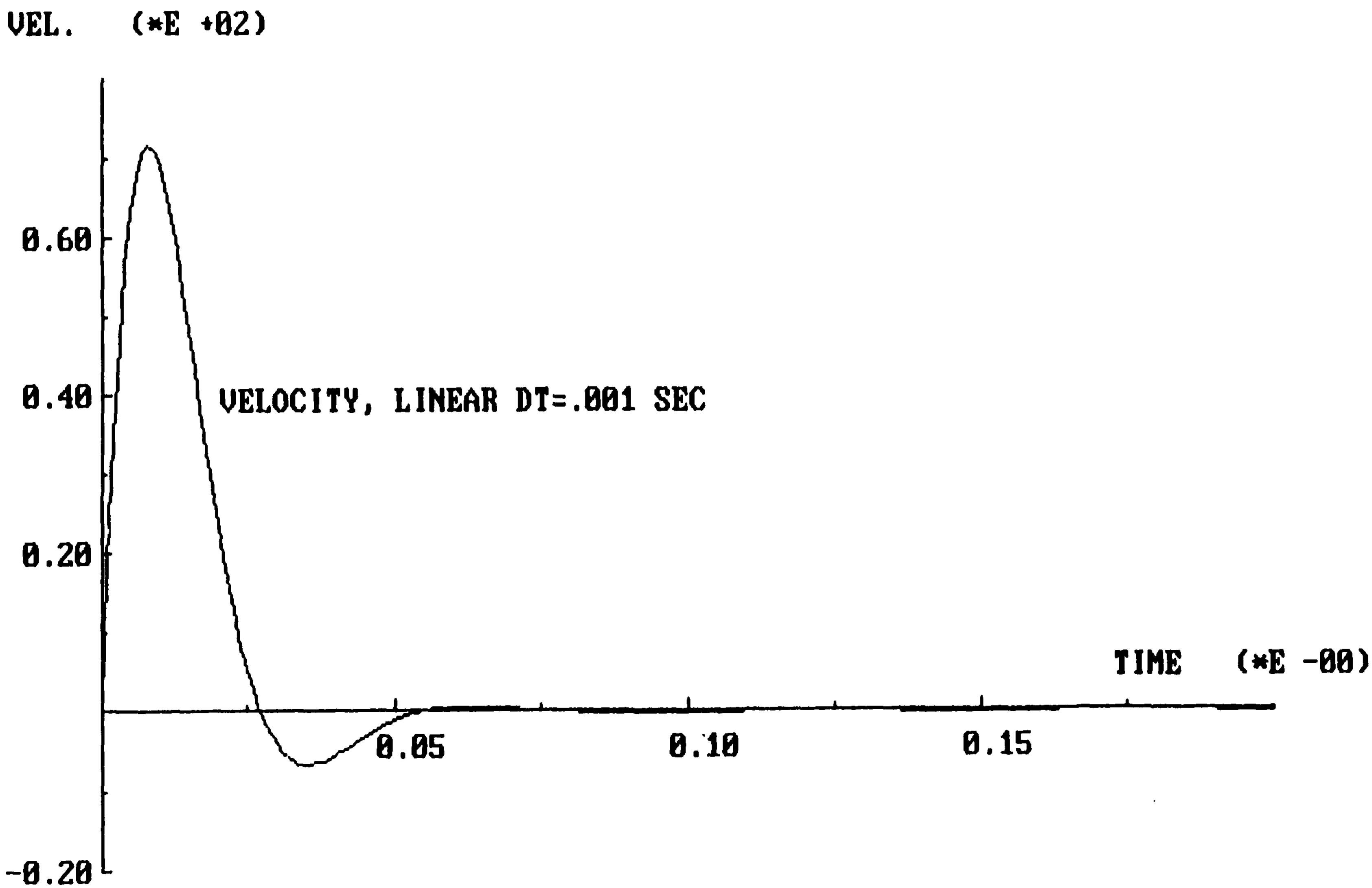


Figure 16. Plot of numerical and analytical solutions to fin velocity vs. time, time step = 0.001 seconds.

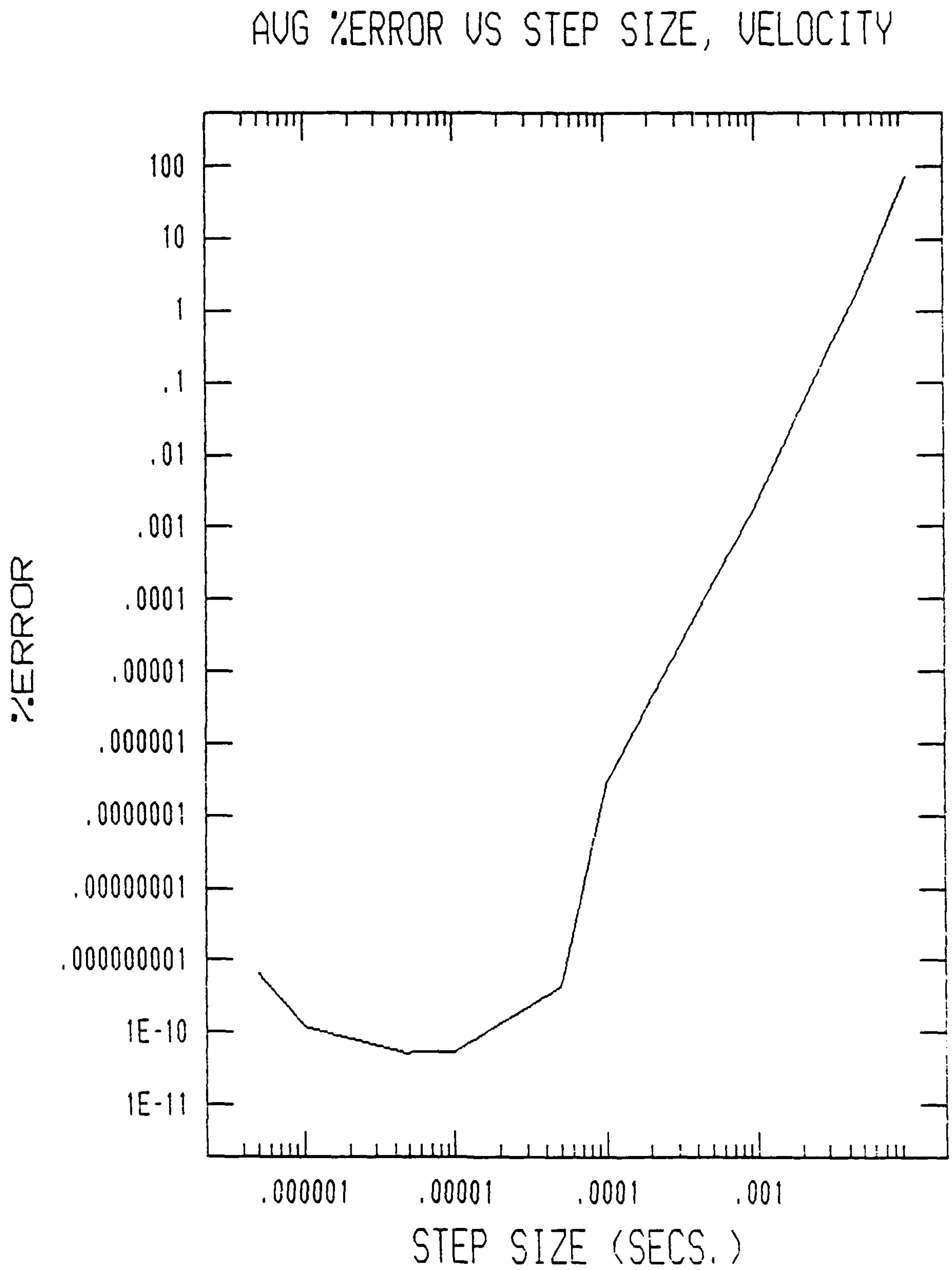


Figure 17. Plot of average percent error vs. time step for first integral.

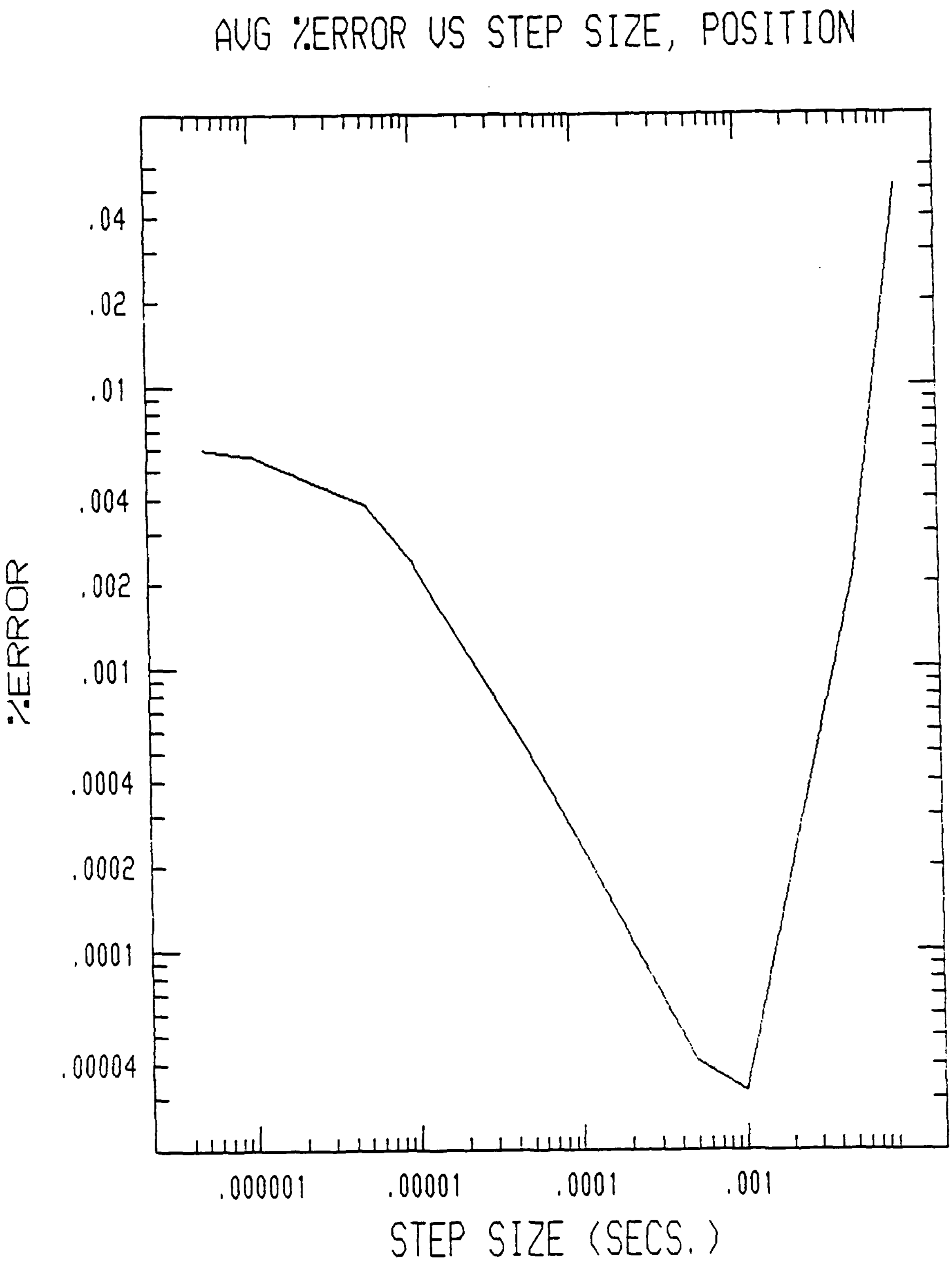


Figure 18. Plot of average percent error vs. time step, 2nd integral.

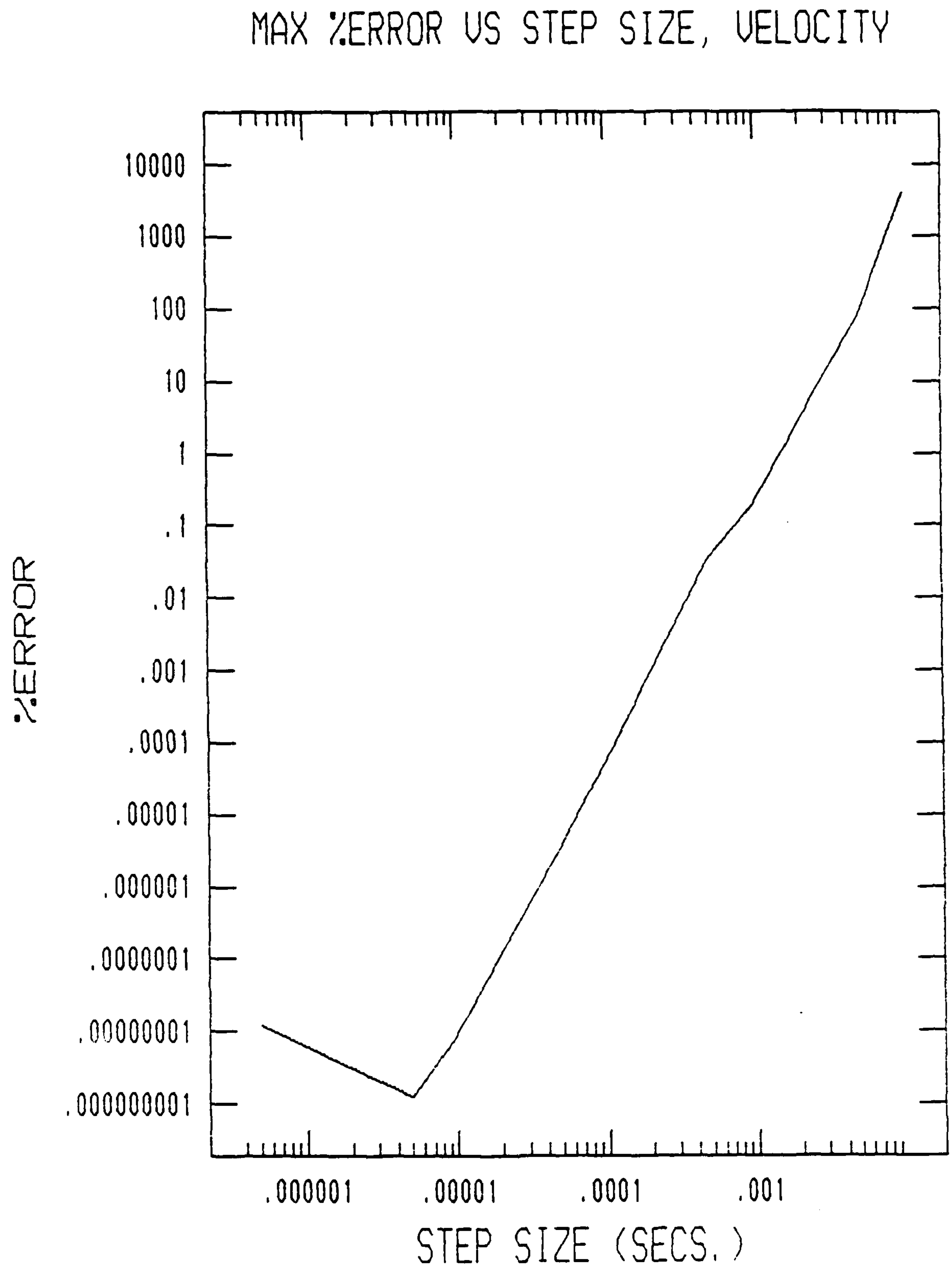


Figure 19. Plot of maximum percent error vs. time step, 1st integral.

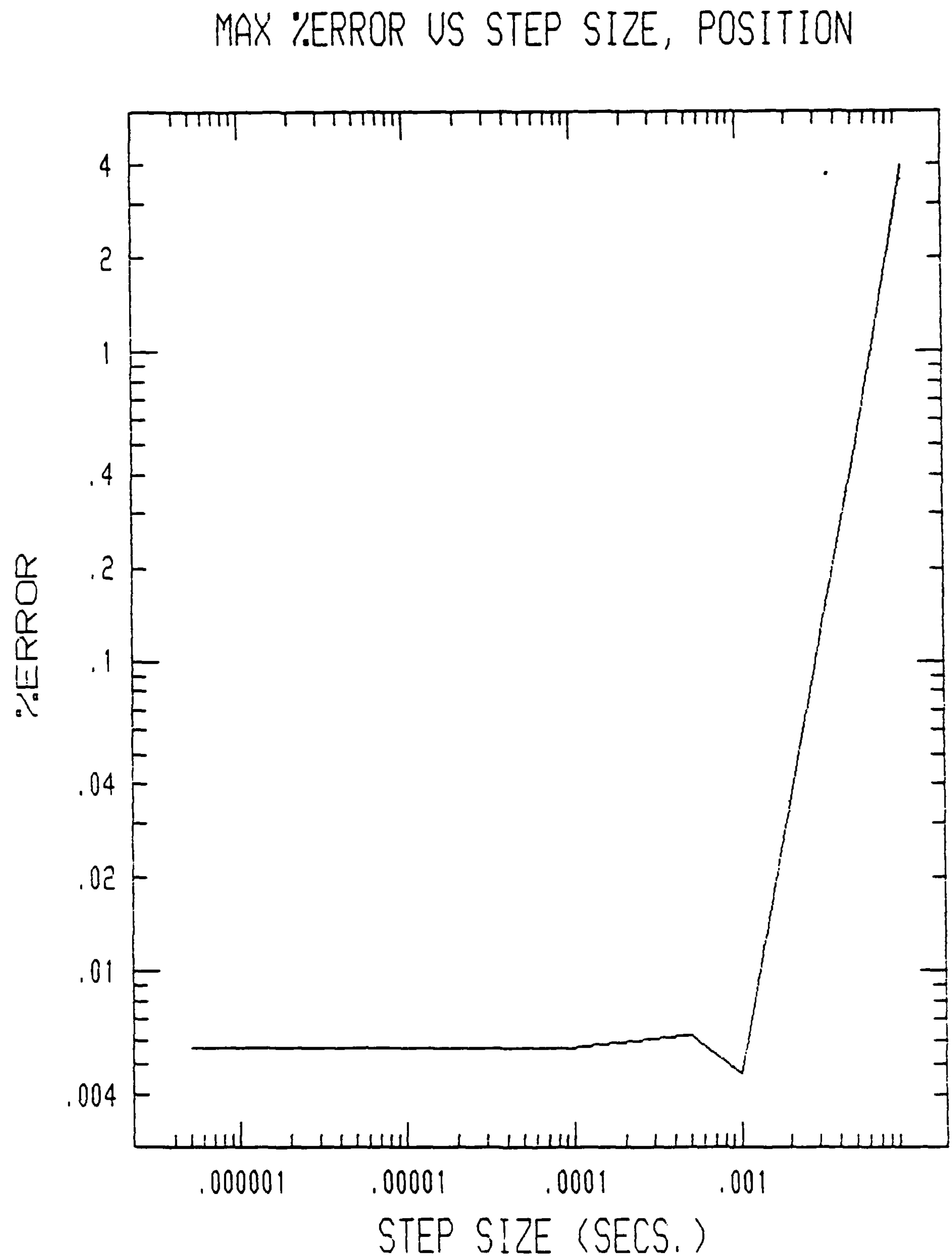


Figure 20. Plot of maximum percent error vs. time step, 2nd integral.

IV. CONCLUSION

The non-linear second-order system more closely modeled the actuator's dynamics and physical characteristics than did the linear system. However, at present we do not have available test data on an actuator's performance and therefore cannot assess the validity of our model. The Runge-Kutta integration scheme gave good accuracy and was deemed dependable. An optimum step size of 0.001 second was chosen from the percent error plots.

REFERENCES

1. Jenkins, Philip N., T-22 Guidance and Navigation Analysis Model, RG-85-12, U.S. Army Missile Laboratory, Redstone Arsenal, AL, January 1985.

2. McKerley, Charles W., Lecture Notes: Missile 6-D Simulation, Southern Institute of Technology, Huntsville, AL, September 1987.

3. Jenkins, Phillip N., An AD-10 5-DOF Autopilot Evaluation and Analysis Model, RD-GC-86-1, Research, Development, and Engineering Center, Redstone Arsenal, AL, December 1985.

APPENDIX

FORTRAN CODE FOR
LINEAR AND NONLINEAR MODELS

APPENDIX

The two versions of the programs for the fin actuator simulation will be given here and the subroutines of interest will be discussed. The linear and non-linear programs are nearly identical except for the subroutine called "ACTUAT", and the analytical solutions in the main program of the linear version. In both versions, two fins of opposite commanded deflections are simulated, but only the results of the positive fin deflection are given since the negative fin deflection gives a mirror image response.

Linear Model

In the linear second-order model, the subroutine "ACTUAT" contains the differential equation that describes the system's dynamics:

```
DEFDD(I)=(OMEGA**2.DO)*(DEFC(I)-DEF(I)-(2.DO*ZETA*DEFD(I)/OMEGA))
```

This equation comes from the differential equation developed earlier. The velocity state (DEFD) is equated to the variable X2 for integration purposes.

The subroutine "ACTINT" initializes the actuator states and describes to the integration scheme which variables are integrals and integrands.

Subroutine "DESOLV" is the Runge-Kutta integration scheme. Variables stored in the array called "IXDOT" are the variables to be integrated. The corresponding location in array "IX" contains the integrated value. In order to perform integrations of orders higher than 1, the result of the first integration is stored as a separate variable name, and this name is submitted to DESOLV for integration. This continues until the proper number of integrations have been performed. This is the reason behind the line of code: DEFD(I) = X2(I). First, DEFDD (acceleration) is integrated and its integral is called "X2". DEFD (velocity) is assigned to the value of X2 and DEFD is integrated to get DEF (position).

Program "DIGSIM" is the calling program and contains the analytical solutions and calculates the errors used in plotting. "C" arrays are used throughout for communication between the various subroutines.

Non-linear Model

The subroutine "ACTUAT" in this version has been modified to include the non-linearities discussed earlier. If the absolute value of the fin velocity is less than 1.6 rads/sec, then the rate feedback is zero. Otherwise it is $\frac{(\dot{\delta} \pm 1.6)}{20}$. This is in addition to the velocity state feedback.

The maximum acceleration allowed is 300 rads/sec^2. The velocity is limited by driving the acceleration to zero as the velocity approaches the slew rate limit of 5.25 rads/sec. The aerodynamic hinge moment is a function of the fin deflection and a hinge moment constant, kHm.

The position limiter sets the velocity and acceleration states to zero and the position state to 0.436 rads if the fin is accelerating past the fin stop.

Note the commanded fin deflection is 0.3 rads. Compared to a unit step (1) command input in the linear model.

The rest of the program for the non-linear model is identical to the linear program, with the exception of the analytical solutions, and are not given.

```
C
      SUBROUTINE ACTUAT
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON C(2000)
      DIMENSION DEF(2),DEFD(2),DEFC(2),X2(2),DEFDD(2)
      EQUIVALENCE (C( 400),DEF(1)),(C( 401),DEF(2)),
     .            (C( 402),DEFD(1)),(C( 403),DEFD(2)),
     .            (C( 404),DEFDD(1)),(C(405),DEFDD(2)),
     .            (C( 406),X2(1)),(C( 407),X2(2)),(C( 1),TIME)
      DATA OMEGA / 144.D0 /
      DATA ZETA / .6D0 /
      DEFC(1) = 1.D0
      DEFC(2) = -1.D0
C      ** ONLY TWO FINS ARE SIMULATED **
      DO 20 I = 1,2
C                *SET DEFD TO X2 FOR INTEGRATION*
      DEFD(I) = X2(I)
C                *ACTUATOR DYNAMICS EQUATION*
      DEFDD(I) = (OMEGA**2.D0)*(DEFC(I)-DEF(I)-(2.D0*ZETA*DEFD(I)/OMEGA))
   20 CONTINUE
      RETURN
      END
C
C     ** THIS SUBROUTINE INITIALIZES THE ACTUATOR STATES **
C     ** AND SETS THE INDEXES FOR THE INTEGRATION SCHEME **
C
      SUBROUTINE ACTINT
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON C(2000)
      COMMON/DEINDX/NDES,IX(100),IXDOT(100)
      DIMENSION DEF(2),DEFD(2),DEFDD(2),X2(2),A(2),ADC(2),ADF(2),
     .          ADDC(2), ADDF(2)
      EQUIVALENCE (C( 400),DEF(1)),(C( 401),DEF(2)),
     .            (C( 402),DEFD(1)),(C( 403),DEFD(2)),
     .            (C( 404),DEFDD(1)),(C(405),DEFDD(2)),
     .            (C( 406),X2(1)),(C( 407),X2(2))
      IX(NDES+1) = 406
      IX(NDES+2) = 407
      IX(NDES+3) = 400
      IX(NDES+4) = 401
      IXDOT(NDES+1) = 404
      IXDOT(NDES+2) = 405
      IXDOT(NDES+3) = 402
      IXDOT(NDES+4) = 403
      NDES = NDES + 4
      DEF(1) = 0.
      DEF(2) = 0.
      DEFD(1) = 0.
      DEFD(2) = 0.
      DEFDD(1) = 0.
      DEFDD(2) = 0.
      X2(1)  = 0.
      X2(2)  = 0.
      RETURN
      END
```

```
C
C       ** MAIN PROGRAM **
C
      PROGRAM DIGSIM
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON C(2000)
      COMMON/DEINDX/NDES,IX(100),IXDOT(100)
      COMMON/FREQ/ N1
      LOGICAL LC(4000),QUIT,DATA
      INTEGER IC(4000)
      DIMENSION DEF(2),DEFD(2)
      EQUIVALENCE (C(1),IC(1))
      EQUIVALENCE (C(1),LC(1))
      EQUIVALENCE (C(    1),TIME   ),(C(   24),TSTOP),(IC(120),NTIME ),
     .            (LC(2046),QUIT),(C(19),DT),(C(408),Y1),(C(409),Y1D),
     .            (C( 410),DIFFR),(C( 411),DIFFRD),(C( 400),DEF(1)),
     .            (C( 402),DEFD(1)),(C( 412),DIFAVG),(C(413),DIFDAVG)
      TSTOP = 100.
    1 CONTINUE
      QUIT=.FALSE.
      CALL INPUT(DATA)
      IF(.NOT.DATA) GO TO 3
C---INITIALIZATION
      TIME  = 0.
      NDES  = 0
      NTIME = 0
      Y1=0.
      Y1D=0.
      COUNT=0.
      DIFSUM=0.
      DIFDSUM=0.
      CALL ACTINT
      CALL ACTUAT
      CALL OUTPUT
C---MAIN LOOP
    2 CONTINUE
      CALL DESOLV
C       ** ANALYTICAL SOLUTION FOR COMPARISON TO NUMERICAL ONE **
      Y1=1.0D0-1.25D0*DEXP(-86.4D0*TIME)*DSIN(115.2D0*TIME+.927295D0)
      Y1D=180.0D0*DEXP(-86.40D0*TIME)*DSIN(115.20D0*TIME)
C         ** DIFFERENCES IN THE ANALYTICAL AND NUMERICAL SOLUTIONS **
      DIFFR =(DEF(1)-Y1)/Y1
      DIFFRD=(DEFD(1)-Y1D)/Y1D
      COUNT=COUNT+1.
      DIFSUM =DIFSUM+DIFFR
      DIFDSUM=DIFDSUM+DIFFRD
      DIFAVG =DIFSUM/COUNT
      DIFDAVG=DIFDSUM/COUNT
      CALL OUTPUT
      CALL TERM
      IF(.NOT. QUIT) GO TO 2
C---POST PROCESSING
      CALL POST1
      CALL OUTPTM
      GO TO 1
    3 CONTINUE
      CALL POST2
      END
```

```
C
C       ** IF THE SIMULATION IS OVER **
C
      SUBROUTINE TERM
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON C(2000)
      EQUIVALENCE (C(1),LC(1))
      LOGICAL LC(4000),QUIT
      EQUIVALENCE (C(   1),TIME   ),(C(  24),TSTOP ),(LC(2046),QUIT)
C
      IF(TIME.GE.TSTOP)THEN
         PRINT *,'SIMULATION ABORT, TIME LIMIT EXCEEDED: TIME = ',TIME
         QUIT = .TRUE.
      ENDIF
      RETURN
      END
C
      BLOCK DATA SIMSUBS
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON C(2000)
      EQUIVALENCE (C(  19),DT     ),(C( 170),HDT    ),(C(  24),TSTOP ),
     .            (C(  27),TFRSP  ),(C(  40),DTFRNT),(C( 114),TFRNTO),
     .            (C( 116),DTFLT  ),(C( 115),TFLTO  ),(C( 711),TFLTSP),
     .            (C( 418),ACTLIM)
      DATA TSTOP / 500. /
      DATA TFRNTO,TFLTO   /  2*0.0     /
      DATA TFRSP,TFLTSP   / 2*10.0     /
      DATA DTFRNT,DTFLT   / 2* 1.0     /
      DATA DT   / .001D0 /
      DATA HDT / .0005D0 /
      END
C
C        ** RUNGE KUTTA INTEGRATION SCHEME **
C
      SUBROUTINE DESOLV
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON C(2000)
      COMMON/DEINDX/NDES,IX(100),IXDOT(100)
      EQUIVALENCE (C(1),IC(1))
      REAL*8 K1(100), K2(100), K3(100), K4(100)
      INTEGER IC(4000)
      DIMENSION XSAVE(100)
      EQUIVALENCE (C(   1),TIME   ),(C(  19),DT     ),(IC(120),NTIME ),
     .            (C( 170),HDT    )
      DATA R6  /.1666666666666666666667D0 /
```

```
*************************************
*  FOURTH ORDER RUNGA-KUTTA METHOD  *
*************************************
C  REVISED VERSION
      DO 1 I1 = 1,NDES
      J = IX(I1)
      XSAVE(I1) = C(J)
    1 CONTINUE
      DO 2 I2 = 1,NDES
      K = IXDOT(I2)
      K1(I2) = DT*C(K)
    2 CONTINUE
      NTIME = NTIME + 1
      TIME = NTIME*HDT
      DO 3 I3 = 1,NDES
      J = IX(I3)
      C(J) = XSAVE(I3) + K1(I3)/2.D0
    3 CONTINUE
      CALL ACTUAT
      DO 4 I4 = 1,NDES
      J = IX(I4)
      K = IXDOT(I4)
      K2(I4) = DT*C(K)
      C(J) = XSAVE(I4) + K2(I4)/2.D0
    4 CONTINUE
      CALL ACTUAT
      DO 5 I5 = 1,NDES
      K = IXDOT(I5)
      K3(I5) = DT*C(K)
    5 CONTINUE
      NTIME = NTIME + 1
      TIME = NTIME*HDT
      DO 6 I6 = 1,NDES
      J = IX(I6)
      C(J) = XSAVE(I6) + K3(I6)
    6 CONTINUE
      CALL ACTUAT
      DO 7 I7 = 1,NDES
      J = IX(I7)
      K = IXDOT(I7)
      K4(I7) = DT*C(K)
      C(J) = XSAVE(I7)+(K1(I7)+2.D0*(K2(I7)+K3(I7))+ K4(I7))/6.D0
    7 CONTINUE
      CALL ACTUAT
      RETURN
      END
```

```
************************************************************************
*    OUTPUT STORES OUTPUT DATA IN THE VARIOUS OUTPUT FILES             *
************************************************************************
      SUBROUTINE OUTPUT
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*4 PC(50)
      INTEGER IC(4000)
      COMMON C(2000)
      EQUIVALENCE (C(1),IC(1))
      CHARACTER NAMES*12
      COMMON/OUTPSC/ NAMES(48)
      COMMON/OUTPTS/ NS,INDEXS(48),NSPD,INDXSP(50)
      EQUIVALENCE (C(   1),TIME   ),(C(  26),TPRNT ),(C(  27),TPRSP ),
     .            (C(  40),DTPRNT),(IC( 206),NO1   ),(C( 114),TPRNTO),
     .            (C( 116),DTPLT ),(IC(1420),NO2   ),(C( 115),TPLTO),
     .            (C( 711),TPLTSP),(C( 712),TPLT   )
C  WRITE PRINT DATA TO LOGICAL UNIT 61 (SCRATCH FILE)
      IF(NS.NE.0) THEN
         IF((TIME.GE.TPRNT).AND.(TIME.LE.TPRSP)) THEN
            WRITE(61)(C(INDEXS(I)),I=1,NS)
            NO1 = NO1 + 1
            TPRNT = TPRNTO + FLOAT(NO1)*DTPRNT
         ENDIF
      ENDIF
C  WRITE PLOT DATA TO LOGICAL UNIT 62
      IF(NSPD.NE.0) THEN
         IF((TIME.GE.TPLT).AND.(TIME.LE.TPLTSP)) THEN
        DO 1 I = 1,NSPD
    1    PC(I) = SNGL(C(INDXSP(I)))
            WRITE(62)(PC(I),I=1,NSPD)
            NO2 = NO2 + 1
            TPLT = TPLTO + FLOAT(NO2)*DTPLT
         ENDIF
      ENDIF
C
      RETURN
      END
************************************************************************
*    OUTPTM STORES THE MULTI RUN DATA IN THE APPROPRIATE OUTPUT FILES    *
************************************************************************
      SUBROUTINE OUTPTM
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*4 PC(50)
      COMMON C(2000)
      CHARACTER NAMEM*12
      COMMON/OUTPMC/ NAMEM(48)
      COMMON/OUTPM/ NM,INDEXM(48),NMPD,INDXMP(50)
C  WRITE PRINT DATA TO LOGICAL UNIT 64 (SCRATCH FILE)
      IF(NM.NE.0) THEN
         WRITE(64)(C(INDEXM(I)),I=1,NM)
      ENDIF
C  WRITE MULTI RUN PLOT DATA TO LOGICAL UNIT 65
      IF(NMPD.NE.0) THEN
        DO 1 I = 1,NMPD
    1    PC(I) = SNGL(C(INDXMP(I)))
        WRITE(65)(PC(I),I=1,NMPD)
      ENDIF
C
      RETURN
      END
```

```
************************************************************************
*  THIS ROUTINE WRITES PRINT DATA IN COLUMN FORMAT GENERATED DURING A RUN  *
************************************************************************
      SUBROUTINE POST1
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON C(2000)
      CHARACTER NAMES*12
      COMMON/OUTPSC/ NAMES(48)
      COMMON/OUTPTS/ NS,INDEXS(48),NSPD,INDXSP(50)
      DIMENSION X(48)
      M=0
    1 CONTINUE
       IF((NS.GT.5*M).AND.(NS.NE.0)) THEN
         M = M+1
         I = 5*(M-1)+1
         J1  = 5*M
         J   = J1
         IF(J1.GT.NS) J=NS
         REWIND(61)
    2    CONTINUE
         WRITE(63,10) (NAMES(K),K=I,J)
C        IF(J1.GT.NS) THEN
C               WRITE(63,*) ' '
C               WRITE(63,*) ' '
C        ENDIF
          DO 3 LINES = 1,501
            READ(61,END=5)(X(K),K=1,NS)
            WRITE(63,20)(X(K),K=I,J)
    3    CONTINUE
C        do 4 1111 = lines,63
C         write(63,*) ' '
C   4    CONTINUE
         GO TO 2
    5    continue
C        do 6 1111 = lines,63
C         write(63,*) ' '
    6    CONTINUE
        go to 1
      ENDIF
      CLOSE(UNIT=61)
      CLOSE(UNIT=62)
      RETURN
   10 FORMAT(5(2X,A12,2X)//)
   20 FORMAT(5(1X,G14.8))
      END
```

```
************************************************************************
*  THIS ROUTINE WRITES PRINT DATA IN COLUMN FORMAT COLLECTED AFTER EACH   *
*  OF A MULT-RUN SET                                                      *
************************************************************************
      SUBROUTINE POST2
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON C(2000)
      CHARACTER NAMEM*12
      COMMON/OUTPMC/ NAMEM(48)
      COMMON/OUTPM/ NM,INDEXM(48),NMPD,INDXMP(50)
      DIMENSION X(48)
      M=0
    1 CONTINUE
      IF((NM.GT.5*M).AND.(NM.NE.0)) THEN
         M = M+1
         I = 5*(M-1)+1
         J1 = 5*M
         J = J1
         IF(J1.GT.NM) J=NM
         REWIND(64)
    2    CONTINUE
         WRITE(66,10)(NAMEM(L),L=I,J)
         IF(J1.GT.NS) THEN
                WRITE(63,*) ' '
                WRITE(63,*) ' '
         ENDIF
         DO 3 LINES = 1,60
            READ(64,END=5)(X(K),K=1,NM)
            WRITE(66,20)(X(K),K=I,J)
    3    CONTINUE
         do 4 llll = lines,63
          write(63,*) ' '
    4    CONTINUE
         GO TO 2
    5    continue
         do 6 llll = lines,63
          write(63,*) ' '
    6    CONTINUE
        go to 1
      ENDIF
      RETURN
   10 FORMAT(1H1/8(2X,A12,2X)//)
   20 FORMAT(8(1X,G14.8))
      END
```

```
C
      SUBROUTINE INPUT(DATA)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON C(2000)
      INTEGER IC(4000)
      LOGICAL LC(4000),DATA,LVAL
      DIMENSION VNAMES(50),VNAMEM(50)
      EQUIVALENCE (C(1),IC(1))
      EQUIVALENCE (C(1),LC(1))
      EQUIVALENCE (C(  26),TPRNT),(C( 114),TPRNTO)
      EQUIVALENCE (IC(206),NO1   ),(IC(1420),NO2    )
      EQUIVALENCE (C( 712),TPLT ),(C( 115),TPLTO )
      character pltfile(50)*8,prntfil(50)*8
      CHARACTER NAMES*12,NAMEM*12,varnam*12,VNAMS2*8
      CHARACTER VNAME*14,VTYPE*1,CARD*80,VNAMES*8,VNAMEM*8
      COMMON/OUTPSC/ NAMES(48)
      COMMON/OUTPTS/ NS,INDEXS(48),NSPD,INDXSP(50)
      COMMON/OUTPMC/ NAMEM(48)
      COMMON/OUTPM/ NM,INDEXM(48),NMPD,INDXMP(50)
      COMMON/FREQ/ N1
      DATA NFIRST / 1  /
      DATA NRUNS/0/
      data prntfil /'prnt1','prnt2','prnt3','prnt4','prnt5','prnt6',
     . 'prnt7','prnt8','prnt9','prnt10','prnt11','prnt12','prnt13',
     . 'prnt14','prnt15','prnt16','prnt17','prnt18','prnt19','prnt20',
     . 'prnt21','prnt22','prnt23','prnt24','prnt25','prnt26','prnt27',
     . 'prnt28','prnt29','prnt30','prnt31','prnt32','prnt33','prnt34',
     . 'prnt35','prnt36','prnt37','prnt38','prnt39','prnt40','prnt41',
     . 'prnt42','prnt43','prnt44','prnt45','prnt46','prnt47','prnt48',
     . 'prnt49','prnt50'/
      data pltfile /'plt1','plt2','plt3','plt4','plt5','plt6',
     . 'plt7','plt8','plt9','plt10','plt11','plt12','plt13',
     . 'plt14','plt15','plt16','plt17','plt18','plt19','plt20',
     . 'plt21','plt22','plt23','plt24','plt25','plt26','plt27',
     . 'plt28','plt29','plt30','plt31','plt32','plt33','plt34',
     . 'plt35','plt36','plt37','plt38','plt39','plt40','plt41',
     . 'plt42','plt43','plt44','plt45','plt46','plt47','plt48',
     . 'plt49','plt50'/
      if(nfirst .eq. 1) THEN
        NS = 0
        NM = 0
        NSPD = 0
        NMPD = 0
      ENDIF
      TPRNT = TPRNTO
      TPLT  = TPLTO
      NO1 = 0
      NO2 = 0
      N1 = 0
      NRUNS = NRUNS + 1
      DATA = .FALSE.
      OPEN(60,FILE='input.DAT',STATUS='OLD')
```

```
C**********************************************************************
C                                                                     *
C     CARD TYPE COLUMNS 1-2                                           *
C       1-INPUT DATA CARD                                             *
C       2-PRINT DATA CARD                                             *
C       3-PLOT DATA CARD                                              *
C       4-INPUT DATA CARD FOR LOGICAL VARIABLES                       *
C       5-DUMP CARD                                                   *
C       9-TERMINATOR CARD                                             *
C       10-COMMENT CARD                                               *
*********************************************************************
**********************************************************************
*     DECODE AN INPUT DATA CARD                                       *
*       DATA IDENTIFIER IN COLUMNS             4-15                   *
*       COMMUNICATION ARRAY INDEX COLUMNS 16-20                       *
*       FORMAT TYPE IN COLUMN                    23                   *
*       FLOATING POINT DATA                    26-40                  *
*       LOGICAL AND INTEGER DATA               26-40                  *
*       ECHO DATA FLAG (0-ECHO, 1-NO ECHO)       42                   *
**********************************************************************
1234  read(60,*,end=999) itype
      backspace 60
      if(itype .gt. 10 .or. itype .lt. 1) go to 543
      goto(100,400,500,543,543,543,543,543,900,1000) itype
 100  READ(60,111) ITYPE,VNAME,INDX,VTYPE
 111  FORMAT(I2,A14,I4,2X,A1)
      if(vtype .eq. 'F' .or. vtype.eq.'f') go to 101
      if(vtype .eq. 'I' .or. vtype.eq.'i') go to 200
      if(vtype .eq. 'L' .or. vtype.eq.'l') go to 300
      print*,'unrecognizable format type'
      go to 1234
 101  backspace 60
      READ(60,10) ITYPE,VNAME,INDX,VTYPE,VALUE,IEDF
      IF(IEDF .NE. 1) THEN
        PRINT*,ITYPE,' ',VNAME,' ',INDX,' ',VTYPE,' ',VALUE
      ENDIF
      C(INDX) = VALUE
 10   FORMAT(I2,A14,I4,2X,A1,E17.8,1X,I1)
      go to 1234
 200  backspace 60
      READ(60,20) ITYPE,VNAME,INDX,VTYPE,IVALUE,IEDF
      IC(INDX) = IVALUE
      IF(IEDF .NE. 1) THEN
        PRINT*,ITYPE,' ',VNAME,' ',INDX,' ',VTYPE,' ',IC(INDX)
      ENDIF
 20   FORMAT(I2,A14,I4,2X,A1,10X,I7,1X,I1)
      GO TO 1234
 300  backspace 60
      READ(60,30) ITYPE,VNAME,INDX,VTYPE,LVAL,IEDF
      LC(INDX) = LVAL
      IF(IEDF .NE. 1) THEN
        PRINT*,ITYPE,' ',VNAME,' ',INDX,' ',VTYPE,' ',LC(INDX)
      ENDIF
 30   FORMAT(I2,A14,I4,2X,A1,2X,L7,9X,I1)
      GO TO 1234
```

```
C***********************************************************************
C      DECODE A PRINT DATA CARD                                        *
C        PRINT HEADER IN COLUMNS                       4-15            *
C        COMMUNICATIONS ARRAY INDEX IN COLUMNS 16-20                   *
C        PRINT DATA COLLECTION FLAG IN COLUMN          23              *
C      MAXIMUM OF 48 PRINT DATA CARDS                                  *
C***********************************************************************
 400   READ(60,40) ITYPE,varnam,IINDX,MPFLG
       backspace 60
       if(mpflg .eq. 1) go to 401
       NS = NS + 1
       IF(NS .GT. 48) THEN
         PRINT*,'TOO MANY PRINT VARIABLES '
         go to 1234
       endif
       READ(60,40) ITYPE,NAMES(NS),INDEXS(NS),MPFLG
 40    FORMAT(I2,1X,A12,1X,I4,2X,I1)
       PRINT*,ITYPE,' ',NAMES(NS),' ',INDEXS(NS),' ',MPFLG
       GO TO 1234
401    NM = NM + 1
       IF(NM .GT. 48) THEN
         PRINT*,'TOO MANY PRINT VARIABLES '
         go to 1234
       ENDIF
       READ(60,40) ITYPE,NAMEM(NM),INDEXM(NM),MPFLG
       PRINT*,ITYPE,' ',NAMEM(NM),' ',INDEXM(NM),' ',MPFLG
       GO TO 1234
C***********************************************************************
C      DECODE PLOT DATA CARDS                                          *
C        PLOT LABEL                          .      4-15               *
C        C ARRAY INDEX COLUMNS                      16-20              *
C        PLOT DATA COLLECTION FLAG                  23                 *
C           THE PLOT DATA COLLECTION FLAG INDICATES WHETHER PLOT DATA IS  *
C           TO BE COLLECTED THROUGHOUT EXECUTION OF A SIMULATION RUN OR   *
C           IS TO BE COLLECTED ONLY AFTER EXECUTION HAS BEEN COMPLETED    *
C      MAXIMUM OF 50 PLOT DATA CARDS                                   *
C***********************************************************************
 500   READ(60,60) ITYPE,VNAMS2,IINDXSP,MPFLG
       backspace 60
       if(mpflg .eq. 1) go to 501
       NSPD = NSPD + 1
       IF(NSPD .GT. 50) THEN
         PRINT*,'TOO MANY PLOT VARIABLES '
         go to 1234
       ENDIF
       READ(60,60) ITYPE,VNAMES(NSPD),INDXSP(NSPD),MPFLG
 60    FORMAT(I2,A8,6X,I4,2X,I1)
       PRINT*,ITYPE,' ',VNAMES(NSPD),' ',INDXSP(NSPD),' ',MPFLG
       GO TO 1234
 501   NMPD = NMPD + 1
       IF(NMPD .GT. 50) THEN
         PRINT*,'TOO MANY PLOT VARIABLES '
         go to 1234
       ENDIF
       READ(60,60)ITYPE,VNAMEM(NMPD),INDXMP(NMPD),MPFLG
       PRINT*,ITYPE,' ',VNAMEM(NMPD),' ',INDXMP(NMPD),' ',MPFLG
       GO TO 1234
 543   read(60,90)card
       print*,'card type not found'
       print*,card
       print*,' '
       go to 1234
1000   read(60,90)card
       print*,card
       go to 1234
```

```
C*********************************************************************************
C      TERMINATOR CARD MUST FOLLOW DATA DECK FOR EACH RUN                       *
C      RUN TITLE IS CONTAINED IN COLUMNS 4-23                                   *
C*********************************************************************************
900    READ(60,90) CARD
       PRINT*,CARD
 90    FORMAT(A)
       IF(NS .NE. 0) THEN
       OPEN(61,FORM='UNFORMATTED',STATUS='SCRATCH')
       OPEN(63,FILE=prntfil(NRUNS),STATUS='UNKNOWN')
       ENDIF
       IF((NM .NE. 0) .AND. (NFIRST .EQ. 1)) THEN
       OPEN(64,FORM='UNFORMATTED',STATUS='SCRATCH')
       OPEN(66,FILE='PRINTM.DAT',STATUS='UNKNOWN')
       ENDIF
       IF(NSPD .NE. 0) THEN
       OPEN(62,FILE=pltfile(nruns),FORM='UNFORMATTED')
       WRITE(62) NSPD,NSPD
       WRITE(62) (VNAMES(III),III=1,NSPD)
       ENDIF
       IF((NFIRST .EQ. 1) .AND. (NMPD .NE. 0)) THEN
       OPEN(65,FILE='PLTM.DAT',FORM='UNFORMATTED')
       WRITE(65) NMPD,NMPD
       WRITE(65) (VNAMEM(III),III=1,NMPD)
       ENDIF
       NFIRST = 0
       DATA = .TRUE.
  999  CONTINUE
       RETURN
       END
```

```
C
      SUBROUTINE ACTUAT
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON C(2000)
      DIMENSION DEF(2),DEFD(2),DEFC(2),X2(2),ACCRQ(2),RATEFB(2),
     .          ACCELIM1(2),ACCELIM2(2),HM(2),DEFDD(2)
      EQUIVALENCE (C( 400),DEF(1)),(C( 401),DEF(2)),
     .             (C( 402),DEFD(1)),(C( 403),DEFD(2)),
     .             (C( 404),DEFDD(1)),(C(405),DEFDD(2)),
     .             (C( 406),X2(1)),(C( 407),X2(2)),(C( 1),TIME)
      DATA RATELM / 5.25D0 /
      DATA OMEGA / 144.D0 /
      DATA ZETA / .6D0 /
      DATA G2 / 300.D0 /
      DATA G1 / 787.5D0 /
      DATA KHM / 0.D0 /
      DEFC(1) = .30D0
      DEFC(2) = -.30D0
C    *** ONLY TWO ACTUATORS ARE SIMULATED ***
      DO 20 I = 1,2
C                    *SET DEFD TO X2 FOR INTEGRATION*
      DEFD(I) = X2(I)
C               *CALCULATE DEADBAND IN RATE FEEDBACK*
      RATEFB(I) = 0.D0
      IF (DEFD(I).GT.1.6D0)  RATEFB(I) = (DEFD(I)-1.6D0)/20.D0
      IF (DEFD(I).LT.-1.6D0) RATEFB(I) = (DEFD(I)+1.6D0)/20.D0
C                    *ACTUATOR DYNAMICS EQUATION*
      ACCRQ(I) = (OMEGA**2.D0)*(DEFC(I)-DEF(I)
     .          -(2.D0*ZETA*DEFD(I)/OMEGA)-RATEFB(I))
C               *CALCULATE SPEED-TORQUE LIMITS*
      ACCELIM1(I) = G1*(1.D0-DEFD(I)/RATELM)
      IF (ACCELIM1(I).GT.G2)  ACCELIM1(I) = G2
      ACCELIM2(I) =-G1*(1.D0+DEFD(I)/RATELM)
      IF (ACCELIM2(I).LT.-G2) ACCELIM2(I) = -G2
      IF (ACCRQ(I).GT.ACCELIM1(I)) ACCRQ(I) = ACCELIM1(I)
      IF (ACCRQ(I).LT.ACCELIM2(I)) ACCRQ(I) = ACCELIM2(I)
C                   *CALCULATE HINGE MOMENTS*
      HM(I) = DEF(I)*KHM
C                   *CHECK FOR POSITION LIMIT*
      IF (DEF(I).GE.0.436D0 .AND. DEFDD(I).GT.0.D0) THEN
          DEFDD(I) = 0.D0
          DEFD(I) =  0.D0
          DEF(I) = 0.436D0
      ENDIF
      IF (DEF(I).LE.-0.436D0 .AND. DEFDD(I).LT.0.D0) THEN
          DEFDD(I) = 0.D0
          DEFD(I) = 0.D0
          DEF(I) = -.436D0
      ENDIF
      DEFD(I) = X2(I)
      DEFDD(I) = ACCRQ(I)-HM(I)
   20 CONTINUE
      RETURN
      END
C
C        ** THIS SUBROUTINE INITIALIZES THE ACTUATOR STATES **
C        ** AND SETS THE INDEXES FOR THE INTEGRATION SCHEME **
C
      SUBROUTINE ACTINT
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON C(2000)
      COMMON/DEINDX/NDES,IX(100),IXDOT(100)
      DIMENSION DEF(2),DEFD(2),DEFDD(2),X2(2),A(2),ADC(2),ADP(2),
     .          ADDC(2), ADDP(2)
```

DISTRIBUTION

	No. of Copies
U.S. Army Materiel System Analysis Activity ATTN: AMXSY-MP Aberdeen Proving Ground, MD 21005	1
IIT Research Institute ATTN: GACIAC 10 W. 35th Street Chicago, IL 60616	1
AMCPM-AT, COL Thomas Kunhart	1
-AT-E, Mr. Bob Lee	1
Mr. Lou Smith	1
AMSMI-RD, Dr. McCorkle	1
Dr. Rhoades	1
Mr. Sliz	1
-RD-GC-N, Mr. McLean	1
Mr. Herbert	1
Mr. Moody	10
-RD-SS, Dr. Grider	1
-RD-GC-C, Mr. Warren	1
-RD-CS-R	15
-RD-CS-T	1
-GC-IP, Mr. Bush	1